Utilisation des nano-composants électroniques dans les architectures de traitement associées aux imageurs

David Roclin

UNIVERSITÉ PARIS-SUD

ÉCOLE DOCTORALE 422 :
SCIENCES ET TECHNOLOGIE DE L'INFORMATION, DES TÉLÉCOMMUNICATIONS ET DES SYSTÈMES

INSTITUT D'ÉLECTRONIQUE FONDAMENTALE
CEA-DRT/LIST/DACLE/LCE

THÈSE DE DOCTORAT

PHYSIQUE

par

David ROCLIN

Utilisation des nano-composants électroniques dans les architectures de traitement associées aux imageurs

Date de soutenance : 16/12/2014

Composition du jury :

Directeur de thèse :	Jacques-Olivier KLEIN	Directeur de Recherche (IEF Paris-Sud)
Encadrants :	Christian GAMRAT	Ingénieur-Chercheur (CEA-DRT/LIST)
	Olivier BICHLER	Ingénieur-Chercheur (CEA-DRT/LIST)
Rapporteurs :	Bertrand GRANADO	Directeur de Recherche (Université Paris 6)
	Jean-Michel PORTAL	Professeur (Université d'Aix-Marseille)
Examinateurs :	Hervé FANET	Ingénieur-Chercheur (CEA-DRT/LETI)
	Lionel TORRES	Directeur de Recherche (Université de Montpellier)

Version du 26 février 2015 (13:47).

Table des matières

Résumé **ix**

Abstract **xi**

Remerciements **xiii**

Introduction **1**

1 Réseaux de neurones et système visuel : Des neurosciences au neuromorphique **7**
1.1 Introduction 7
1.2 Les neurones impulsionnels 8
1.2.1 Le neurone biologique 8
1.2.2 Le neurone LIF 9
1.3 Perception de l'information 11
1.3.1 La rétine biologique 11
1.3.2 Les Rétines artificielles 13
1.4 Le cortex visuel 14
1.4.1 Organisation du cortex 14
1.4.2 Modèle de cortex, le modèle HMAX 15
1.4.3 Les réseaux convolutionnels 16
1.5 Méthode d'apprentissage 18
1.5.1 STDP biologique 18
1.5.2 Apprentissage non-supervisé avec STDP 19
1.5.3 STDP simplifiée 21
1.6 Systèmes neuromorphiques dédiés à la vision 21
1.6.1 Première génération et approche classique 21
1.6.2 Réseaux impulsionnels 22
1.7 Discussion et perspectives 25

2 Optimisation d'un réseau de neurones impulsionnels **27**
2.1 Introduction 27
2.2 Application Détection de véhicule 29
2.3 Simulateur XNET 30
2.4 Impact de la résolution des poids synaptiques. 31
2.5 Impact d'une décroissance linéaire du potentiel membranaire 34
2.6 Impact du remplacement des fenêtres temporelles par une FIFO 36
2.7 Résultat comprenant l'ensemble des modifications 38
2.8 Mises en place des optimisations 39
2.9 Discussion et perspectives 40

3 Synapses et Nanotechnologies : Mémoires émergentes 43
3.1 Introduction . . . 43
3.2 Les dispositifs memristifs . . . 44
3.2.1 Introduction . . . 44
3.2.2 Les familles de dispositifs memristifs . . . 45
3.3 Le dispositif memristif : la synapse idéale ? . . . 46
3.3.1 Période de rétention . . . 47
3.3.2 Résolution du poids synaptique . . . 47
3.3.3 Cycle de programmations . . . 49
3.3.4 Densité d'intégration . . . 49
3.3.5 Méthode de programmation . . . 50
3.4 Discussion et perspectives . . . 51

4 Mémoire synaptique à base de dispositifs memristifs 53
4.1 Introduction . . . 53
4.2 Modélisation VHDL-AMS d'une cellule CBRAM . . . 55
4.2.1 Modèle CBRAM . . . 55
4.2.2 Mémoire synaptique à base de dispositifs memristifs CBRAM . . . 57
4.3 Structure de CBRAM en crosspoint . . . 58
4.3.1 Crossbar . . . 59
4.3.2 Matrice connectée par les anodes . . . 65
4.3.3 Matrice connectée par les cathodes . . . 69
4.3.4 Matrice – Surcoût associé au transistor de sélection dans une matrice . . . 70
4.3.5 Conclusions des études réalisées . . . 70
4.4 Modélisation d'un réseau de neurones impulsionnels et simulation VHDL-AMS . . . 71
4.4.1 Réseau de neurones impulsionnels en VHDL-AMS . . . 71
4.4.2 Simulation du réseau . . . 75
4.5 Discussion et perspectives . . . 76

5 Co-intégration CMOS/CBRAM dédiée au Neuromorphique 79
5.1 Introduction . . . 79
5.2 Choix des structures et des entrées/sorties . . . 80
5.2.1 Modes de programmation et de lecture . . . 81
5.2.2 Choix des entrées/sorties . . . 82
5.3 Description de l'architecture . . . 84
5.3.1 Architecture des circuits d'interface . . . 84
5.3.2 Architecture haut niveau . . . 85
5.3.3 Architecture du crossbar . . . 86
5.3.4 Architecture de la matrice . . . 86
5.3.5 Caractéristique du layout du circuit . . . 87
5.4 Environnement de test . . . 89
5.4.1 Circuit Imprimé . . . 89
5.4.2 FPGA . . . 90
5.5 Phases de tests . . . 90
5.5.1 Test du circuit sous Eldo . . . 91
5.5.2 Validation du circuit imprimé et du FPGA . . . 91
5.5.3 Test du circuit . . . 92
5.6 Discussion et perspectives . . . 94

6 Conclusion et perspectives **97**
6.1 Conclusion 97
6.2 Perspectives 98

Annexes **101**

Publications personnelles **109**

Bibliographie **111**

Table des figures

1 Tendance des capteurs d'image CMOS 1
2 Exemple de réseau de neurones impulsionnels 4

1.1 Description d'un neurone 8
1.2 Potentiel d'action et synapse 9
1.3 Modèle de neurone impulsionnel 10
1.4 Principe de fonctionnement d'un réseau impulsionnel 11
1.5 Système de vision chez l'humain 12
1.6 Rétine AER – Description du pixel et principe de fonctionnement 13
1.7 Organisation du cortex 14
1.8 Vitesse de perception visuelle chez le singe 15
1.9 Représentation du modèle HMAX. (Riesenhuber and Poggio (1999)) . . 15
1.10 Invariance obtenue par les cellules complexes 16
1.12 Noyaux de convolutions après apprentissage 17
1.13 Fenêtre temporelle STDP relevé par Bi et Poo (Bi and Poo (1998)) . . 19
1.14 Fenêtre temporelle de la STDP biologique 19
1.15 Apprentissage STDP sur des visages et des motos 20
1.16 STDP simplifiée 21
1.17 Historique des implémentations matérielles de réseaux de neurones . . . 22
1.18 CAVIAR : Présentation du projet 23
1.19 CAVIAR : Exemple de noyaux de convolutions 23
1.20 CAVIAR : Reconstruction des évènements en sortie des différentes puces 24
1.21 CAVIAR : Activité des neurones en simulation 25

2.1 Etude d'un réseau de neurones impulsionnels 28
2.2 Architecture du réseau impulsionnel de détection de véhicules 29
2.3 Calcul des erreurs dans XNET 31
2.4 Simulation - Résolution synaptique 32
2.5 Simulation - Résolution synaptique - Paramètres communs 33
2.6 Simulation – Fuite linéaire 35
2.7 Simulation – STDP par ordre 38
2.8 Reconstruction des poids synaptique 39
2.9 Simulation – toutes modifications appliquées 40

3.1 Les quatre composants passifs fondamentaux en électronique 44
3.3 Memristor et synapse 46
3.4 Impact de la résolution synaptique 47
3.5 Évolution de la capacité des mémoires non-volatile 49
3.6 Relation Tension-Courant de dispositifs unipolaires et bipolaires 50
3.7 Méthode de programmation des dispositifs unipolaires et bipolaires . . . 51

4.1 Réseau impulsionnel à base de dispositifs memristifs 54
4.2 Cycle de programmation et de lecture du réseau 54

4.3 Image TEM d'un dispositif CBRAM 55
4.4 Modélisation d'un dispositif CBRAM 56
4.5 Effet de la limitation de courant lors d'un SET 57
4.6 Programmation stochastique extrinsèque 58
4.7 Extraction des valeurs de résistances parasites 59
4.8 Crossbar – Consommation d'énergie 60
4.9 CBRAM – Valeurs de résistance LRS avec différentes tensions de programmation 61
4.10 Crossbar - Modélisation pendant un SET 61
4.11 Crossbar – chute de tension 63
4.12 CBRAM – valeur de résistance LRS en fonction de la limitation de courant 64
4.13 Crossbar – Courant de lecture 65
4.14 Crossbar – Perturbation des cellules voisines 66
4.15 Description des structures en matrice 66
4.16 Matrice connectée par les anodes – Consommation d'énergie 67
4.17 Matrice connectée par les anodes – Modélisation pendant un SET . . . 67
4.18 Matrice connectée par les anodes – Chutes de tension 68
4.19 Matrice connectée par les anodes – Courant de lecture 69
4.20 Matrice connectée par les cathodes – Consommation d'énergie 70
4.21 Matrice – Coûts silicium 71
4.22 Topologie du réseau simulé en VHDL-AMS 72
4.23 Programmation stochastique SET et RESET 73
4.24 Structure des neurones 74
4.25 Automate du neurone d'entrée 74
4.26 Automate du neurone de sortie 75
4.27 Résultat de la simulation VHDL-AMS 77

5.1 Chronologie de fabrication et tests du circuit 80
5.2 Architecture haut niveau du circuit 80
5.3 Condition de programmation et de lecture du crossbar 81
5.4 Condition de programmation – interaction de pulses 82
5.5 Condition de programmation et de lecture de la matrice 82
5.6 Brochage du circuit 83
5.7 Schéma de la porte de transmission implémentant un circuit d'interface. 84
5.8 Schéma du circuit translateur de niveau logique. 85
5.9 Layout a) d'un montage élévateur de tension et b) d'une porte de transmission. 85
5.10 Architecture du circuit crossbar 86
5.11 Architecture des circuits d'interface du crossbar 87
5.12 Architecture du circuit de la matrice 87
5.13 Architecture des circuits d'interface de la matrice 88
5.14 Layout du crossbar et de la matrice 88
5.15 Circuit imprimé de test avec le circuit neuromorphique au centre. 89
5.16 Eldo : simulation des circuit 1R et 1T1R 91
5.17 Eldo : simulation du crossbar 92
5.18 Temps d'impulsion SET-RESET à l'oscilloscope 93
5.19 Forming d'une CBRAM 94

1 PCB de test pour la puce neuromorphique. Les principaux composant sont : 5 DAC, 1 ADC, et 16 convertisseur de niveaux. 106
2 PCB de test pour la puce neuromorphique. Les principaux composant sont : 5 DAC, 1 ADC, et 16 convertisseur de niveaux 107

Liste des tableaux

1.1 Meilleurs résultats du benchmark MNIST 18

2.1 Liste des paramètres des synapses et des neurones dans XNET 30
2.2 Valeurs des paramètres de la simulation figure 2.4 32
2.3 Valeurs des paramètres de la simulation figure 2.5 33
2.4 Valeurs des paramètres de la simulation figure 2.6 35
2.5 Valeur des paramètres de l'évolution génétique pour la STDP par ordre 37
2.6 Valeur des paramètres de la simulation STDP par ordre figure 2.7 . . . 37
2.7 Nombre moyen de synapses renforcées 38
2.8 Valeurs des paramètres de la simulation figure 2.9 39

3.1 Ordre de grandeur de la résolution synaptique en fonction du mode d'apprentissage et de l'application. 48
3.2 Tableau comparatif de la taille des cellules mémoires non-volatiles . . . 50
3.3 Tableau comparatif des technologies mémoires non-volatiles 51

4.1 Paramètres physiques du modèle VHDL-AMS de la CBRAM. 56
4.2 Paramètres neuronaux et synaptiques de la simulation VHDL-AMS. . . 75

5.1 Plots analogique – Sept entrées et une sortie. 83
5.2 Plots de sélections des anodes et cathodes. 83
5.3 Table de vérité du signal $SELECT < 1..0 >$: sélection des topologies. . 86
5.4 Table de vérité du signal $PAD\_MUX < 1..0 >$: sélection de la cathode à lire. 86
5.5 Condition de programmation des résultats présenté figure 5.20. 94

1 Table de vérité de la logique contrôlant les portes de transmission d'une anode et d'une cathode dans le crossbar. 103
2 Table de vérité de la logique contrôlant les portes de transmission d'une anode, d'une cathode et d'une ligne de grilles dans la matrice. 103

Résumé

En utilisant les méthodes d'apprentissages tirées des récentes découvertes en neuroscience, les réseaux de neurones impulsionnels ont démontrés leurs capacités à analyser efficacement les grandes quantités d'informations provenant de notre environnement. L'implémentation de ces circuits à l'aide de processeurs classiques ne permet pas d'exploiter efficacement leur parallélisme. L'utilisation de mémoire numérique pour implémenter les poids synaptique ne permet pas la lecture ou la programmation parallèle des synapses et est limité par la bande passante reliant la mémoire à l'unité de calcul. Les technologies mémoire de type memristive pourrait permettre l'implémentation de ce parallélisme au cœur de la mémoire.

Dans cette thèse, nous envisageons le développement d'un réseau de neurones impulsionnels dédié au monde de l'embarqué à base de dispositif mémoire émergents. Dans un premier temps, nous avons analysé un réseau impulsionnel afin d'optimiser ses différentes composantes : neurone, synapse et méthode d'apprentissage STDP en vue d'une implémentation numérique. Dans un second temps, nous envisageons l'implémentation de la mémoire synaptique par des dispositifs memristifs. Enfin, nous présentons le développement d'une puce co-intégrant des neurones implémentés en CMOS avec des synapses en technologie CBRAM.

Mots clefs : memristor, dispositif memristif, réseau de neurones impulsionnels, système neuromorphique, spike-timing-dependent plasticity.

Abstract

By using learning mechanisms extracted from recent discoveries in neuroscience, spiking neural networks have demonstrated their ability to efficiently analyze the large amount of data from our environment. The implementation of such circuits on conventional processors does not allow the efficient exploitation of their parallelism. The use of digital memory to implement the synaptic weight does not allow the parallel reading or the parallel programming of the synapses and it is limited by the bandwidth of the connection between the memory and the processing unit. Emergent memristive memory technologies could allow implementing this parallelism directly in the heart of the memory.

In this thesis, we consider the development of an embedded spiking neural network based on emerging memory devices. First, we analyze a spiking network to optimize its different components: the neuron, the synapse and the STDP learning mechanism for digital implementation. Then, we consider implementing the synaptic memory with emergent memristive devices. Finally, we present the development of a neuromorphic chip co-integrating CMOS neurons with CBRAM synapses.

Keywords: memristor, memristive device, Spiking Neural Networks, Neuromorphic architecture, spike-timing-dependent plasticity.

Remerciements

Je tiens tout d'abord à remercier mon Directeur de thèse, Jacques-Olivier Klein, Professeur à l'Institut d'Electronique Fondamentale (IEF), Paris Sud XI, ainsi que Christian Gamrat, Ingénieur Chercheur au CEA LIST pour m'avoir fait confiance en me permettant de réaliser cette thèse. Leurs encadrements, leurs conseils et nos échanges réguliers ont permis de faire progresser efficacement mes travaux.

Je remercie Thierry Colette, Chef du DACLE, pour son accueil et pour m'avoir fourni les moyens de réaliser cette thèse. Je remercie également Raphaël David, Chef du LCE et Yannick Bonhomme, Chef du LFSE pour leur accueil au sein de leur laboratoire ainsi que leur confiance et soutien.

Je remercie Bertrand Granado, Directeur de Recherche à l'Université Paris 6, et Jean-Michel Portal, Professeur à l'Université d'Aix-Marseille, pour avoir accepté de juger mon travail en tant que rapporteurs ainsi que pour leur participation à mon jury de thèse. Je remercie également Hervé Fanet, Ingénieur-Chercheur au CEA LETI, pour sa participation à mon jury de thèse et Lionel Torres, Directeur de Recherche à l'Université de Montpellier, pour avoir accepté de présider mon jury de thèse.

Je remercie aussi Simon Thorpe, Directeur de recherche au Centre national de la recherche scientifique (CNRS), pour nos fructueuses conversations qui m'ont grandement aidé à démarrer mes travaux.

Je tiens aussi à remercier mes collègues du LCE et du LFSE, et notamment mes collègues de bureau, Alexandre Carbon, Julien Bezine, Marie Krumpe Goldsztejn, Hong Phuc Trinh, Raphael Karabassis et Andrea Castagnetti pour la bonne ambiance et le soutien apporté pendant ces trois années de thèse.

Je remercie mes collègues de l'IEF, Damien Querlioz et Weisheng Zhao, pour leur contribution ainsi que mes collègues du CEA Grenoble, notamment des laboratoires DCOS et LISAN pour leur collaboration me permettant ainsi de parvenir au succès lors de la réalisation du circuit neuromorphique.

Un remerciement tout particulier à mon collègue et ami, Olivier Bichler, qui a su m'encourager pendant ces trois années et partager avec moi ses expériences et son vaste savoir. J'ai sincèrement apprécié nos discussions animées aussi bien sur le neuromorphique que sur celles de la vie quotidienne.

Enfin, je tiens à remercier ma famille pour son soutien et son dévouement durant ces longues années d'études et surtout mon épouse qui a su supporter mes soirées et week-ends d'écriture et de préparation. Elle a toujours su trouver les mots pour m'encourager et m'accompagner dans cette belle aventure.

Introduction

Grâce aux avancés de la technologie micro/nano-électronique, notamment dans le domaine de la miniaturisation des composants, l'industrie tend à incorporer un nombre important de capteurs dans les objets de la vie quotidienne comme nos voitures et nos téléphones (internet des objets). Les capteurs photographiques, les capteurs olfactifs ou les capteurs acoustiques sont des exemples de capteurs de données naturelles, c'est-à-dire de données de notre environnement naturel issues de capteurs. Ces capteurs, composés principalement de transistors MOS, suivent la loi de Moore ce qui permet d'atteindre de plus grandes densités d'intégration (Figure 1). Les capteurs photographiques embarqués dans nos téléphones possèdent des résolutions pouvant atteindre 41 Mégapixels. De plus, certaines applications (notamment de sécurité) exigent des fréquences d'acquisition d'images supérieures à 120 images par seconde. De grandes résolutions à de hautes fréquences génèrent de grandes quantités de données. Bien qu'il ne soit généralement pas nécessaire d'effectuer une analyse automatique de ces acquisitions pour des applications de la vie quotidienne, il en est différemment pour des applications de sécurité. La détection de piétons, la reconnaissance d'individus ou la perception d'un environnement pour une application robotique exigent l'analyse en temps réel et en local de la totalité des informations acquises par ces capteurs. Or, les architectures et algorithmes de traitement actuels capables de traiter ces données (CPU-GPU) ne sont pas adaptés au monde de l'embarqué où superficie et consommation d'énergie sont des facteurs primordiaux.

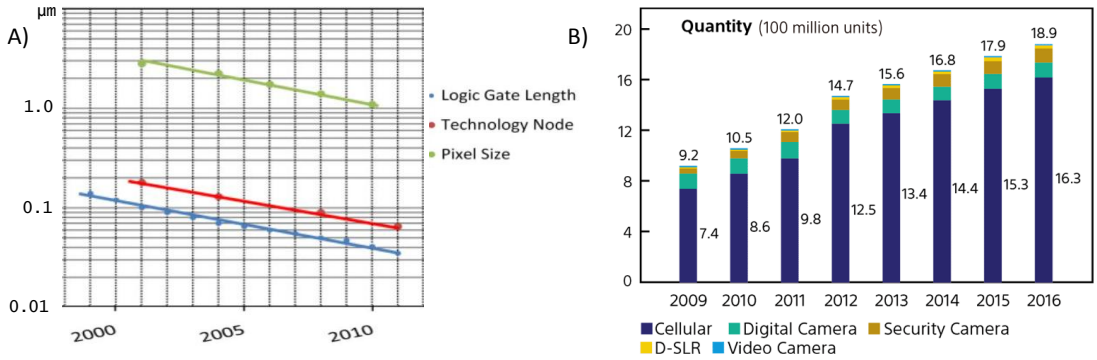


FIGURE 1 – à gauche : tendance des capteurs d'image CMOS (Hwang (2012)), en bleu, longueur minimale de la porte logique (ITRS (2013)), en rouge, nœud technologique utilisé, en vert, taille minimale d'un pixel en μm. B) Estimation Sony du marché des capteurs d'image «Complementary Metal Oxide Semiconductor» (CMOS) en millions d'unités (Sony (2012))

Dans la recherche d'architecture novatrice capable de traiter ces données, un organe biologique fascine : le cerveau. En effet, la nature a créé et fait évoluer cet organe entièrement dédié au traitement de données naturelles. Notre cerveau est capable d'analyser des informations provenant de millions de récepteurs sensoriels (ex : photorécepteurs, environ 125 millions de cellules photosensibles par œil). Le cerveau traite ces informations localement, en parallèle, avec un temps de réponse minimum et ne dissipe qu'une

faible puissance (<20W). La première génération de réseaux de neurones artificiels (exemples : perceptrons, MLP (MultiLayer Perceptron), réseau de Hopfield et machine de Boltzman (Anderson and Rosenfeld (2000))) modélise l'absence ou la présence d'une impulsion et est basée sur des neurones binaires à seuil (Siu et al. (1995)). La seconde génération n'est plus binaire, elle modélise la fréquence des impulsions en utilisant une représentation multi-valuée et est basée sur des neurones à sigmoïde (Bishop (1995)) . Ces générations ont montré qu'elles étaient très bien adaptées aux traitements de données naturelles massives. Pourtant elles restent éloignées de la biologie et des récentes découvertes en neuroscience, notamment dans le codage de l'information. En effet, bien que la fréquence des impulsions joue un rôle important dans les aires sensoriel primaires, des travaux ont montré que le cerveau peut réaliser des taches complexes en environ 150 ms (Thorpe et al. (1996)) avec un chemin comprenant seulement 10 couches de traitements. Or, les fréquences impulsionnelles enregistrées sur ces couches sont inferieures à 100 Hz, ce qui est contradictoire avec un codage purement fréquentiel si l'on considère un chemin visuel de 10 couches parcouru en 150 ms. En effet, des travaux ont montré que certain neurones utilisent plutôt le temps précis des impulsions pour coder l'information (Rieke et al. (1999)), ce qui a donné lieu à la troisième génération de réseaux de neurones artificiels.

Les Réseaux de Neurones Impulsionnels (RNI) ou Spiking Neural Networks (SNN) représentent la troisième génération de réseaux de neurones artificiels. Cette génération se base sur des modèles de neurones impulsionnels. Ces neurones émettent des potentiels d'action (ou impulsions électriques) vers d'autres neurones lorsque leur seuil d'activation est atteint. Ces réseaux utilisent donc des impulsions électriques pour transporter l'information et les récents progrès en neurosciences suggèrent que l'information est encodée dans les délais entre les impulsions, c'est un codage relatif (Rieke et al. (1999)). Ce codage plus riche de l'information a permis de mettre en valeur la plasticité synaptique connue sous le nom de Spike-Timing-Dependant-Plasticity (STDP). Cette règle d'apprentissage permet la modification de l'efficacité de la connexion entre deux neurones en fonction de leurs activités grâce notamment à deux mécanismes, la potentialisation à long terme (LTP) et la dépression à long terme (LTD). Ce mécanisme d'apprentissage a démontré son efficacité à analyser les données naturelles et son adaptation aux variations de l'environnement (Song et al. (2000); Indiveri (2002); Arthur and Boahen (2005); Liu (2003); Bichler et al. (2012a)). Chaque neurone est en effet capable de traiter les potentiels d'action et d'appliquer la règle d'apprentissage STDP de façon autonome créant ainsi un vaste champ d'unités de calcul pouvant s'exécuter de façon asynchrone et parallèle. Les réseaux de neurones impulsionnels ont montré des performances supérieures aux réseaux de première et seconde générations Maass (1997) mais restent privés d'architectures embarquables dédiées spécifiquement à leur nature parallèle et asynchrone malgré quelques efforts (Indiveri et al. (2006); Schemmel et al. (2010)).

En effet, ces réseaux sont généralement simulés sur des architectures de type Von-Neumann. Alors que les algorithmes classiques ont été développés et optimisés pour ce type d'architecture (instructions séquentielles), l'implémentation de réseaux de neurones massivement parallèle possédant un grand nombre de paramètres synaptiques n'est pas optimale sur ce type d'architecture séquentielle. Le cœur de traitement de cette architecture, le processeur (CPU - Central Processing Unit) récupère une petite quantité d'information dans la mémoire, effectue un calcul avec ces informations puis enregistre le résultat dans cette même mémoire (fetch-decode-execute-store) de façon séquentielle alors que l'algorithme neuronal est par essence totalement parallèle. Bien que cette architecture ait démontré ses capacités sur des algorithmes traditionnels, l'augmentation de la quantité d'information perçue par ces capteurs ainsi que le grand

nombre de poids synaptiques se heurte à ce que l'on appelle le goulot d'étranglement de Von-Neumann (Von-Neumann bottleneck). En effet, un seul bus relie la mémoire où sont stockées les informations et les unités de calcul qui les traitent. Ce bus à une bande passante maximum, ce qui restreint la quantité d'information qui peut y circuler. Cela limite le nombre de calculs pouvant être effectués en parallèle dans les unités de calcul. La simulation d'un réseau de neurones sur ce type d'architecture n'est pas extensible à de très larges réseaux. Pour atteindre des réseaux de grande taille et de faible consommation, il est donc indispensable de créer des architectures VLSI dédiées spécifiquement aux réseaux de neurones impulsionnels.

Depuis l'introduction du terme « neuromorphique » par Carver Mead dans les années 80, les scientifiques tentent d'implémenter des réseaux de neurones avec des systèmes VLSI dédiés. Dans ce domaine, deux orientations de recherche peuvent être discerné. Dans la première orientation, le neuromimétique, l'objectif est de reproduire, au plus proche la biologie, un cerveau humain (ou seulement une partie) afin de le simuler ou de l'émuler. Plusieurs équipes ont déjà proposé des implémentations matérielles dédiées (projet BrainScaleS, projet Neurogrid). Dans la seconde orientation, le neuromorphique, celle qui nous intéresse dans cette thèse, l'objectif est de créer des architectures neuronales dédiées et embarquables, des systèmes possédant donc une faible surface de silicium et une consommation d'énergie à l'échelle de celle du cerveau.

Le point faible des systèmes neuromorphiques généralement proposés est la séparation entre la mémoire et l'unité de calcul ainsi que l'utilisation de mémoire classique dont le fonctionnement (lecture/écriture) est séquentielle. Dans les réseaux de neurones biologiques, les mécanismes d'apprentissage s'effectuent directement au niveau des synapses et en parallèle. Le récent développement de nouveaux dispositifs mémoires pourrait permettre l'implémentation des mécanismes d'apprentissage de la STDP directement au cœur de la mémoire.

En effet, de récents travaux ont permis de développer des dispositifs mémoires de type memristif (Strukov et al. (2008); Kund et al. (2005); Raoux et al. (2008); Wong et al. (2010a, 2012)). Ces dispositifs à deux terminaux ont la capacité de modifier leur conductance quand une tension est appliquée à leurs bornes, laissant entrevoir la possible d'implémenter des mécanismes de plasticité synaptique sur ces dispositifs, utilisés comme des synapses. Pour des réseaux impulsionnels, la conductance d'un dispositif peut implémenter un poids synaptique. Ces dispositifs possèdent une faible énergie de commutation (de l'ordre du pJ par bit) (ITRS (2013)). Leur arrangement en crossbar (Figure 2) permet d'obtenir une forte densité d'intégration (100 nm^2 par dispositif (ITRS (2013))) mais surtout la possibilité d'implémenter le parallélisme naturel des réseaux de neurones. En effet, les structures en crossbar permettent de 1) lire l'ensemble des poids synaptiques d'un même neurone d'entrée en parallèle et de 2) programmer les synapses d'un même neurone de sortie en parallèle (Figure 2).

Dans cette thèse, nous envisageons le développement d'un réseau de neurones impulsionnels dédié au monde de l'embarqué à base de dispositifs mémoires émergents. Dans un premier temps, nous avons analysé un réseau impulsionnel afin d'optimiser ses différentes composantes : neurone, synapse et méthode d'apprentissage STDP en vue d'en faciliter et de rendre réaliste son implémentation. Dans un second temps, nous envisageons l'utilisation de dispositifs memristifs pour réaliser la mémoire synaptique dans un tel réseau. Enfin, nous présentons le développement d'une puce co-intégrant des neurones implémentés en CMOS avec des synapses en technologie CBRAM.

Organisation du mémoire Le mémoire est organisé de la manière suivante :

– Le premier chapitre, «Réseaux de neurones et système visuel : Des neurosciences au neuromorphique», introduit les réseaux de neurones ainsi que le chemin de

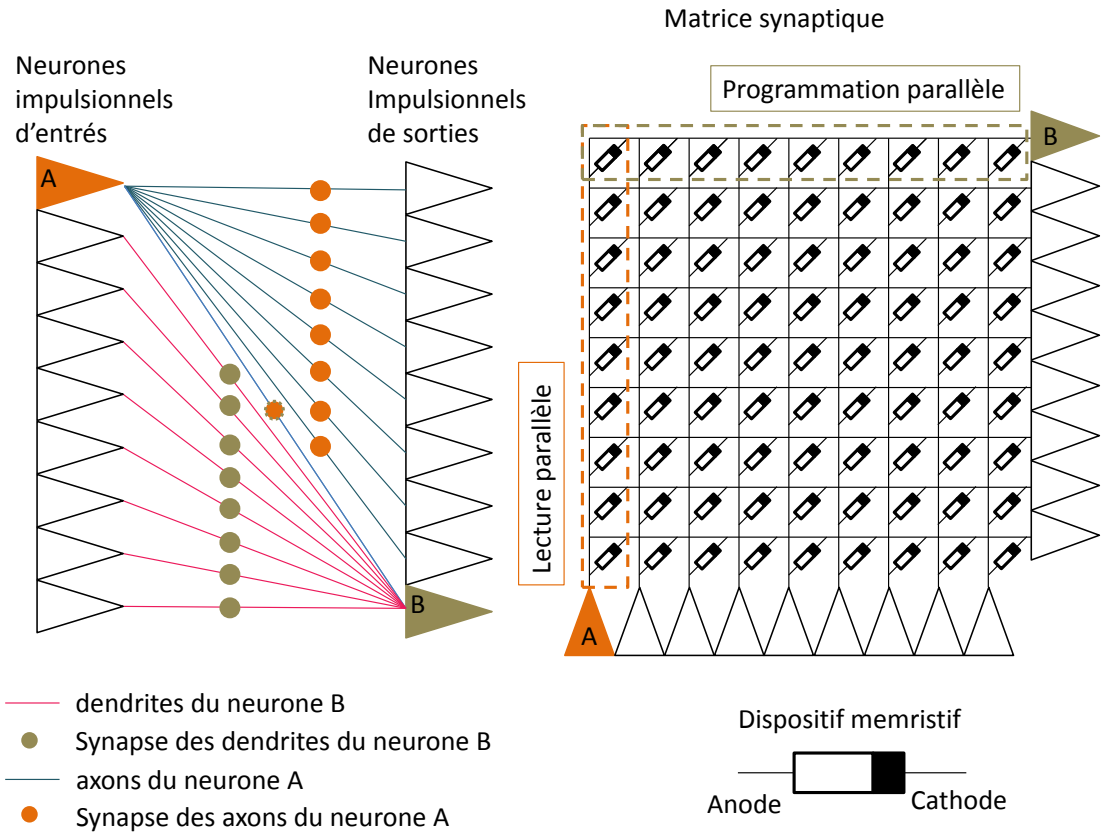


FIGURE 2 – Réseau de neurones impulsionnel avec 10 neurones d'entrée et 10 neurones de sortie. Grâce à la structure en crossbar, la lecture et la programmation des synapses peuvent être effectuées en parallèle.

l'information visuelle dans le cerveau. Pour cela, nous abordons dans un premier temps ces sujets du point de vue des neurosciences, et dans un second temps, leur modèle neuromorphique correspondant. Puis, nous établissons un état de l'art des implémentations neuromorphiques dédiées à la vision.

– Le second chapitre, «Optimisation d'un réseau de neurones impulsionnels», présente les travaux publiés lors de la conférence IJCNN 2013 (International Joint Conference on Neural Networks). Dans ce chapitre, nous nous intéressons à l'implémentation optimisée d'un réseau impulsionnel pour tenter de répondre aux fortes contraintes caractéristiques au monde de l'embarqué, la superficie et la consommation d'énergie. Pour cela, nous cherchons tout d'abord à identifier les éléments du réseau impulsionnel pouvant limiter l'architecture en termes de superficie, de ressource calculatoire, et de consommation d'énergie. Nous en avons identifié trois, la résolution des poids synaptiques, le modèle du neurone et en particulier sa composante de fuite, et le mécanisme d'apprentissage STDP. Enfin, nous allons voir comment optimiser ces éléments pour atteindre un gain de superficie et de consommation afin de rendre ce type de réseau compatible au monde de l'embarqué, qui ont été validés dans la réalisation d'un démonstrateur dans un stage que j'ai co-encadré.
– Le troisième chapitre, «Synapses et Nanotechnologies : Mémoires émergentes», introduit et présente un état de l'art des dispositifs memristifs. Nous évaluons ces dispositifs afin de savoir s'ils pourraient permettre d'implémenter de manière efficace les synapses d'un réseau impulsionnel. Pour cela, nous identifions les critères essentiels recherchés lors de la création d'une synapse artificielle. Puis, nous comparerons les différentes technologies memristives sur ces différents critères.

- Le quatrième chapitre, «Mémoire synaptique à base de dispositifs memristifs», présente les travaux publiés lors de la conférence NanoArch 2014 (International Symposium on Nanoscale Architectures). Dans ce chapitre, nous étudions différentes structures synaptiques, le crossbar et les matrices, pour les comparer sur différents critères comme la consommation d'énergie, les chutes de tension, la variation du courant de lecture et la perturbation des cellules voisines. Pour cela, nous définissons le modèle VHDL-AMS d'un dispositif memristif, la CBRAM, puis nous présentons des simulations VHDL-AMS des structures proposées. Ces résultats nous permettent de définir la structure répondant le mieux à nos besoins lors de la création d'une mémoire synaptique à base de dispositifs memristifs.
- Le cinquième chapitre, «Co-intégration CMOS/CBRAM dédiée au Neuromorphique», présente le développement du premier circuit neuromorphique intégrant un procédé CMOS classique 130nm avec une technologie memristive CBRAM. Nous présentons l'environnement des tests pratiqués et les résultats validant ce circuit neuromorphique.
- Enfin, le chapitre de conclusion va nous permettre d'établir une synthèse des travaux de la thèse ainsi que d'ouvrir sur des orientations de travaux futurs.

Chapitre 1

Réseaux de neurones et système visuel : Des neurosciences au neuromorphique

Sommaire

1.1 Introduction 7
1.2 Les neurones impulsionnels 8
1.2.1 Le neurone biologique 8
1.2.2 Le neurone LIF 9
1.3 Perception de l'information 11
1.3.1 La rétine biologique 11
1.3.2 Les Rétines artificielles 13
1.4 Le cortex visuel 14
1.4.1 Organisation du cortex 14
1.4.2 Modèle de cortex, le modèle HMAX 15
1.4.3 Les réseaux convolutionnels 16
1.5 Méthode d'apprentissage 18
1.5.1 STDP biologique 18
1.5.2 Apprentissage non-supervisé avec STDP 19
1.5.3 STDP simplifiée 21
1.6 Systèmes neuromorphiques dédiés à la vision 21
1.6.1 Première génération et approche classique 21
1.6.2 Réseaux impulsionnels 22
1.7 Discussion et perspectives 25

1.1 Introduction

Dans ce premier chapitre d'état de l'art, nous allons mettre en relation les informations tirées des découvertes en neuroscience et en perception humaine avec leur implémentation neuromorphique dans l'optique de pouvoir réaliser des architectures capable de traiter efficacement les données naturelles issues de capteurs.

Dans un premier temps, nous allons décrire le fonctionnement du neurone biologique et l'un de ses modèles fonctionnels, le modèle « LIF ». Ensuite, nous étudierons comment notre système de vision convertit la lumière en influx nerveux et présenter un exemple de rétine artificielle. Puis, nous allons comprendre comment les neurones s'organisent dans le cortex visuel pour recréer des informations complexes (formes, visages) et présenter

deux modèles de cortex visuels. Nous allons ensuite comprendre les mécanismes liés à l'apprentissage et leurs implémentations. Enfin nous allons décrire un exemple de système de vision neuromorphique complet, le système CAVIAR.

1.2 Les neurones impulsionnels

1.2.1 Le neurone biologique

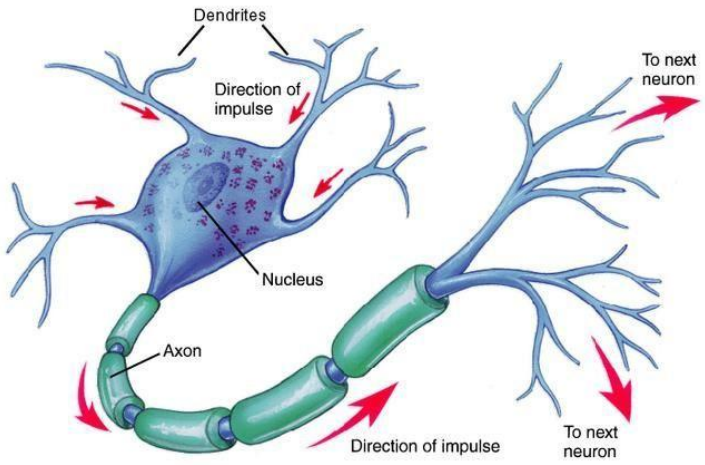


FIGURE 1.1 – Les potentiels d'actions sont propagés des dendrites vers le soma. Quand le soma génère un potentiel d'action, il voyage à travers l'axone et toutes ses ramifications vers les synapses d'autres neurones. (H (2014))

La structure de base du système nerveux central est le neurone. Il est présent chez l'individu adulte au nombre moyen de 100 milliards (10^{11}). Son rôle est de générer des impulsions électriques en réponse à une activité chimique et de les transporter vers d'autres neurones, de manière rapide, parfois sur de longues distances. Un neurone est constitué d'un corps cellulaire, appelé soma, et de deux types de prolongement : en entrée, les dendrites et en sortie, un axone. Les dendrites, de l'ordre de 10^3 par neurone, transportent les potentiels d'actions provenant d'autres neurones vers le soma. Un unique axone, dont la terminaison se ramifie (arborisation terminale), transporte le potentiel d'action du soma vers d'autres neurones. L'axone d'un neurone A est connecté à la dendrite d'un neurone B par une synapse. Chaque neurone peut posséder entre 1 et 100 000 synapses pour un nombre total de synapses pour un individu adulte d'environ 100 à 500 trillions (10^{14}) Purves (2008).

Le signal électrique de référence du neurone est la différence de potentiel entre l'intérieur de la cellule neuronale et son extérieur. En repos, le potentiel électrique intracellulaire est généralement -70 mV comparé au potentiel extracellulaire (communément défini à 0V). Cette tension électrique, ou potentiel membranaire, est maintenu grâce à des canaux ioniques présents dans la membrane. À la réception d'un potentiel d'action pré-synaptique (impulsion du neurone précédant une de ses synapses, un neurone d'entrée), le potentiel membranaire entre en phase d'hypopolarisation et se voit augmenter (moins négatif). Si ce potentiel membranaire atteint un certain seuil ($\approx -55mV$), une forte dépolarisation s'effectue, un potentiel d'action est alors généré. Ces impulsions électriques (potentiel d'action) émises par les neurones ont une amplitude d'environ 100 mV sur une durée de l'ordre de la milliseconde. Cette phase de dépolarisation est suivie d'une phase d'hyperpolarisation, le potentiel membranaire devient très négatif

($\approx -80mV$). Durant cette période, même avec de fortes activités en entrée, le neurone est quasiment dans l'incapacité de générer un second potentiel avant un certain temps ($\approx 0.1ms$). Cette période est appelée période réfractaire. Elle permet notamment de limiter le nombre d'activations du neurone lors de la présence d'un motif en entrée et donc de neutraliser une information qui serait redondante.

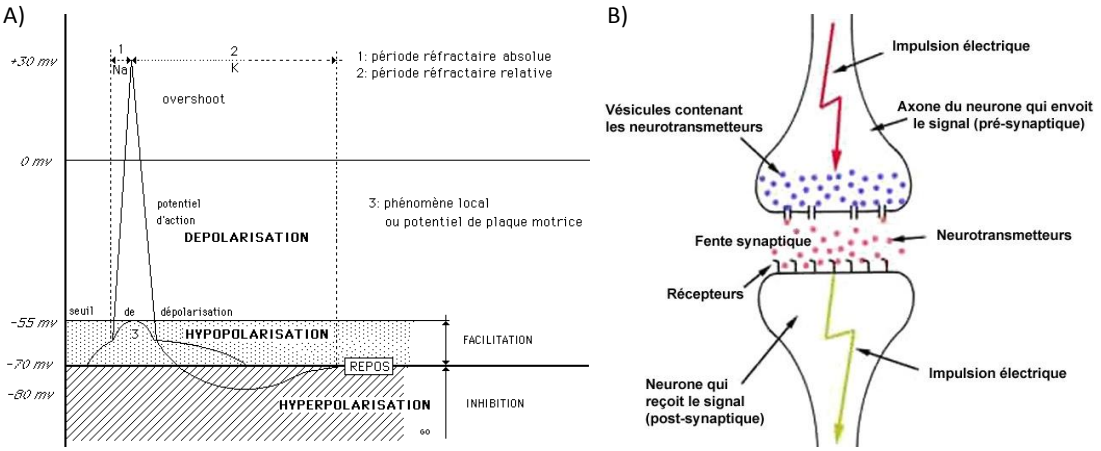


FIGURE 1.2 – A) Dans cet exemple, le potentiel d'action effectue une forte dépolarisation suivie d'une faible hyperpolarisation. (Chevrier (2014)) B) Schéma de la jonction synaptique (Déthiollaz (2014))

L'arrivée de ce potentiel d'action à la terminaison de l'axone, au niveau de la synapse, produit un flux d'ions Ca^{2+}, et déclenche une libération de neurotransmetteurs. Ces neurotransmetteurs se fixent à des récepteurs de l'autre côté de la jonction synaptique (dendrite du neurone post-synaptique) et ouvrent des canaux conducteurs d'ions (figure 1.2 B). Ces courants d'ions peuvent alors avoir un effet excitateur (dépolarisation) ou inhibiteur (hyperpolarisation) sur le neurone post-synaptique (Dayan and Abbott (2005)).

1.2.2 Le neurone LIF

Durant ma thèse, nous nous sommes exclusivement consacrés à la troisième et dernière génération de réseaux de neurone artificiel, les réseaux impulsionnels. Comparés aux deux générations précédentes basées sur des portes à seuil ou à sigmoïde (McCulloch and Pitts (1943),Rosenblatt (1988)), ces réseaux, tirés directement des découvertes en neuroscience, modélisent précisément les neurones et leurs interactions synaptiques. Ils utilisent des impulsions asynchrones pour communiquer et peuvent donc tirer avantage d'une nouvelle dimension, le temps entre les impulsions. Ces réseaux sont composés de neurones impulsionnels qui peuvent être modélisés selon plusieurs degrés de complexité. Dans sa forme la plus complexe, le modèle Hodgkin-Huxley (Hodgkin and Huxley (1952)), les canaux ioniques sont modélisés et notamment les canaux des ions Potassium (K) et Sodium (Na) (figure 1.3 A)). L'équation 1.1a régit ce modèle, où I est le courant membranaire, E est le potentiel membranaire, C_M est la capacité membranaire, R_K et R_{Na} représentent la résistance du Potassium et du sodium, E_K et E_{Na} les potentiels de repos des ions Potassium et calcium, et R_l et E_l la résistance et le potentiel de repos de la fuite. Des modèles simplifiés en sont dérivés, par exemple le modèle FitzHugh-Nagumo (Fitzhugh (1961); Nagumo et al. (1962); Linares-Barranco et al. (1991)).

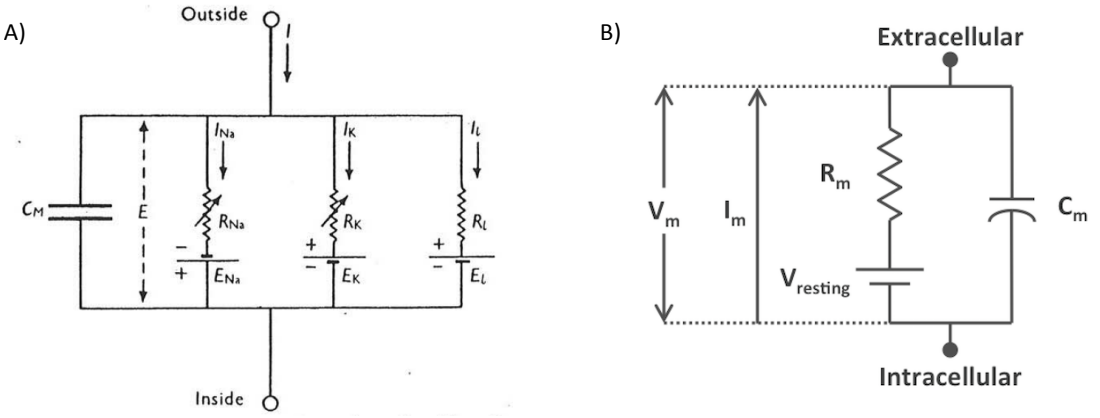


FIGURE 1.3 – A) Modèle de neurone Hodgkin-Huxley. Les trois « R » représentent la résistance des différents canaux ioniques. Le condensateur C_M représente la capacité du potentiel membranaire et E le potentiel membranaire. Les flèches sur des canaux K et Na indiquent une résistance variable en fonction de la tension appliquée. Les générateurs E forcent les potentiels de repos (Hodgkin and Huxley (1952)) B) Modèle de neurone Leaky Integrate and Fire (Kurniawan (2014))

$$I = C_M \frac{dE}{dt} + \frac{E - E_K}{R_K} + \frac{E - E_{Na}}{R_{Na}} + \frac{E - E_l}{R_l} \tag{1.1a}$$

$$I = C_m \frac{dV_m}{dt} + \frac{V_m - V_{resting}}{R_m} \tag{1.1b}$$

Dans nos travaux, nous utilisons le modèle dans sa forme la plus simple présenté équation 1.1b, où V_m est le potentiel membranaire, C_m est la capacité membranaire, R_m la résistance membranaire et $V_{resting}$ le potentiel membranaire au repos. En électronique, ce modèle s'implémente tout simplement avec un condensateur modélisant le potentiel membranaire, en parallèle avec une résistance (figure 1.3 B)). Sa simplicité permet une meilleure intégration pour une implémentation dans un système embarqué dont les performances ont été démontrées, le modèle « Leaky Integrate and Fire ». En effet, ce type de neurone effectue les trois actions élémentaires d'un neurone impulsionnel, il « intègre » les courants des impulsions pré-synaptiques pondérées par les synapses, mais cette intégration « fuit » exponentiellement. Et lorsque celle-ci atteint un seuil donné, le neurone émet une impulsion vers les neurones des couches postérieures.

Comme les neurones biologiques, les neurones LIF possèdent une période réfractaire et d'inhibition latérale. La période réfractaire, due à une hyperpolarisation après un potentiel d'actions, empêche un neurone de s'activer plusieurs fois de suite dans une période de temps très court (de l'ordre de la milliseconde) et permet une remise à zéro de l'intégration du neurone. La période d'inhibition latérale permet à un neurone, lors de son activation, de désactiver les neurones présents sur la même couche que lui-même, en forçant une remise à zéro de leur intégration et en les empêchant d'intégrer de nouveau potentiel d'actions. En biologie, ce mécanisme est effectué grâce à des connexions synaptiques inhibitrices. Il permet à un neurone de se spécialiser et ainsi éviter que plusieurs neurones apprennent et répondent à un même stimulus d'entrées. Dans les travaux présentés chapitre 2, les neurones d'entrées du réseau (comme ceux présents figure 1.3 B)) sont activés par le flux AER provenant de la camera DVS128. En effet, un neurone d'entrée correspond à un évènement (ON ou OFF) d'un pixel de la caméra. Quand un pixel émet un évènement, il émet son adresse (x,y) ainsi que le type d'évènement et active son neurone d'entrée correspondant. Sachant que la résolution

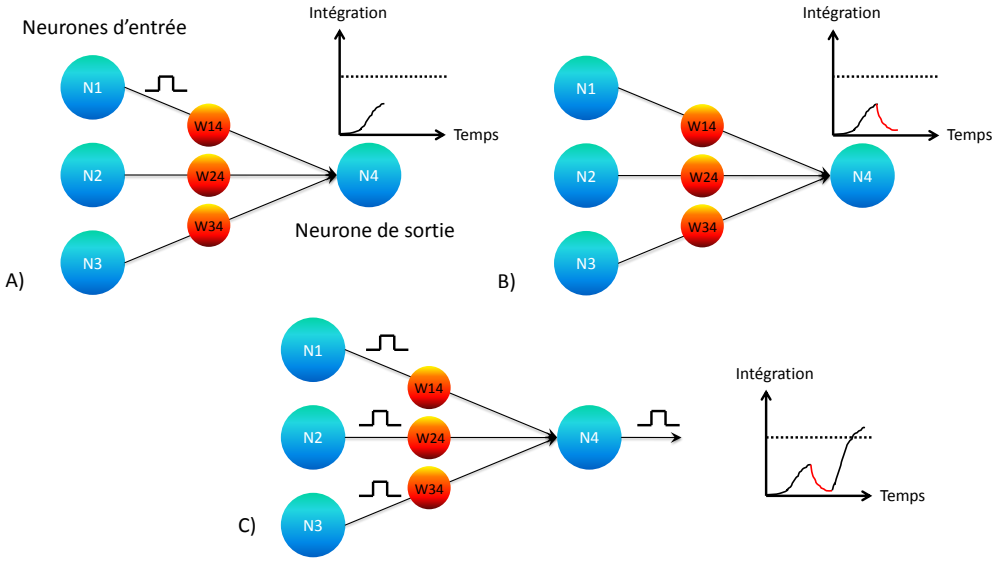


FIGURE 1.4 – A) Le neurone intègre une impulsion du neurone présynaptique N1 pondéré par la synapse W14. B) L'intégration du neurone fuit exponentiellement. C) Avec de fortes activités en entrée, l'intégration du neurone atteint un seuil donné et émet une impulsion vers les couches postérieures

de la caméra est de 128×128, ces réseaux possèdent $128 \times 128 \times 2$ neurones d'entrées, soit 32 768.

1.3 Perception de l'information

1.3.1 La rétine biologique

Notre organe de vision, l'œil (figure 1.5 a)), est composé de cellules photoréceptrices capables de transformer les ondes électromagnétiques en impulsions électriques. Ce sont des cellules nerveuses contenant des substances chimiques sensibles à la lumière, les photopigments, qui déclenchent une réponse électrique. Ces cellules existent sous deux formes, les cônes et les bâtonnets (Figure 1.5 b)). Les cônes représentent environ 5% des cellules photoréceptrices (environ 5 millions de cônes) alors que les bâtonnets représentent 95% des cellules photoréceptrices d'un individu adulte (environ 100-120 millions de bâtonnets) (Curcio et al. (1990); Schacter et al. (2011)).

Les bâtonnets sont présents sur la périphérie de la rétine, mais moins dans la partie centrale de la rétine, la fovéa. Ils sont responsables de la vision nocturne (vision scotopique) et sont plus sensibles à de faibles luminosités que les cônes.

Les cônes sont majoritairement présents dans la fovéa. Ils sont plus actifs la journée, car responsables de la vision diurne (vision photopique) et plus particulièrement de la vision des couleurs. L'œil humain possède trois types de cônes (trichromatisme), les cônes cyanolabes, chlorolabes, et érytholabes, tous sensibles à une gamme de longueurs d'onde différente, 420–440 nm (bleu), 534–545 nm (vert) et 564–580 nm (rouge) respectivement. Les cônes ont une meilleure résolution spatiale que les bâtonnets.

Les photorécepteurs, reliés entre eux par les cellules horizontales (Figure 1.5 c)), sont ensuite reliés à des cellules bipolaires ON et OFF. À la présentation d'un faisceau lumineux, les cellules ON ont une activation graduée et les cellules OFF une inactivation graduée. Ces cellules, connectées les unes aux autres par des cellules amacrines, sont ensuite connectées aux cellules ganglionnaires pour créer des champs récepteurs. Les cellules ganglionnaires sont les seules cellules de la rétine à pouvoir générer des potentiels d'actions, ou impulsions électriques, et sont au nombre d'un million chez

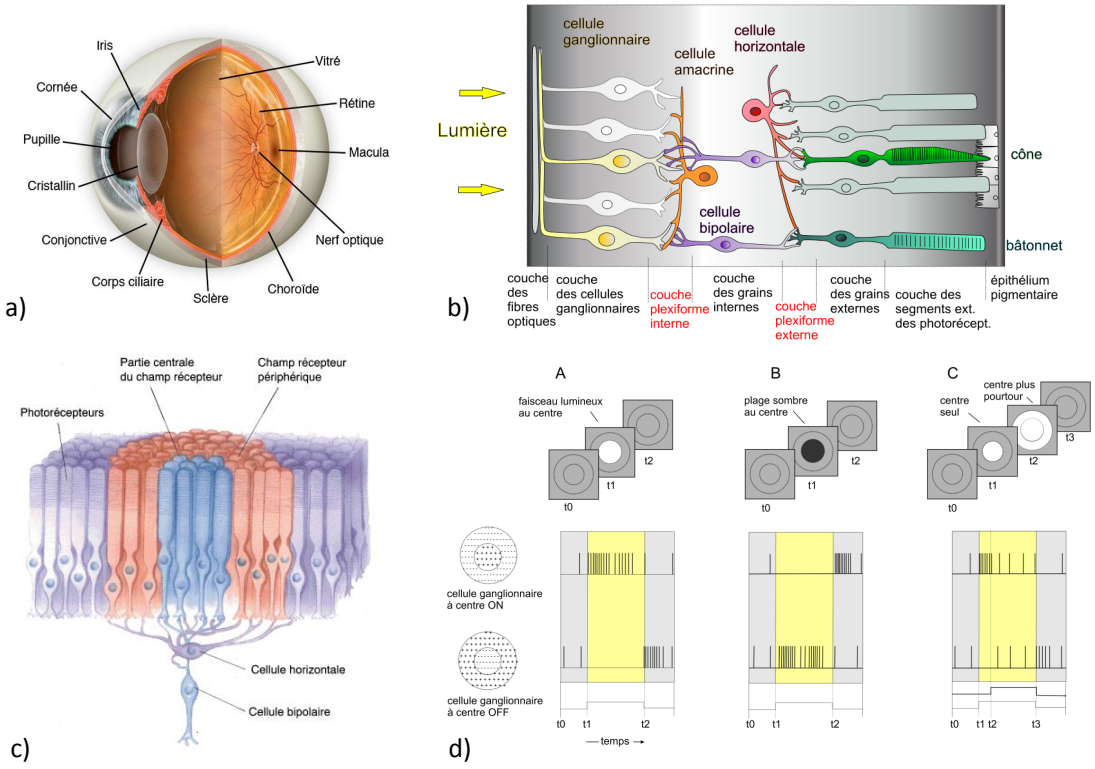


FIGURE 1.5 – a) Œil humain, les photorécepteurs sont situés dans la rétine, sur la partie arrière de l'œil. (ima (2014)) b) Coupe transversale de la rétine. Les cônes et bâtonnets, situés sur la partie arrière de la rétine, sont connectés par des synapses aux cellules bipolaires ON OFF, elles-mêmes connectées aux cellules ganglionnaires. Enfin, le nerf optique, situé sur la partie avant de la rétine rassemble les informations des cellules ganglionnaires sous forme de potentiels d'actions pour les envoyer dans le cortex visuel. (Purves and Coquery (2005)) c) Les cellules horizontales connectent les cônes et les bâtonnets ensemble pour créer des champs récepteurs. (Bear et al. (2007)) d) La fréquence des potentiels s d'actions en sortie des cellules ganglionnaires est proportionnelle à la luminosité appliquée sur leur champ récepteur. (Purves (2008))

les humains. L'activité en terme de potentiels d'actions des cellules ganglionnaires est proportionnelle à la luminosité appliquée sur leur champ récepteur (Figure 1.5 d)). Les cellules ganglionnaires envoient directement les potentiels d'actions vers le cortex, via leurs axones qui forment le nerf optique, pour poursuivre le traitement des informations.

Afin de créer des capteurs d'images, intelligent et embarquable, proches des performances de l'œil humain, nous pouvons à ce moment constater quelques caractéristiques intéressantes de cet organe :

- Contrairement à un capteur d'image CMOS traditionnel, il n'y a pas de notion de trames. Les bâtonnets et cônes, pouvant être considérés comme les « pixels » de l'œil, peuvent s'activer indépendamment d'une manière « asynchrone ».
- L'œil ne possède pas qu'un seul type de pixel, mais plutôt quatre, dont trois pour les couleurs et un pour le contraste. Il serait donc intéressant d'implémenter ces quatre types de pixels sur une rétine artificielle.
- Une impulsion électrique est produite seulement en cas de variations de la luminosité, donc lors de la modification du stimulus visuel. Cela permet de ne pas introduire d'informations redondantes, contrairement au système à base de trames.
- La résolution n'est pas uniforme. Elle est supérieure au centre de l'image et inférieure au bord de l'image. On peut donc imaginer un capteur dont la résolution

diminuerait en s'éloignant de son centre ou avec une résolution fixe intégrant un subsampling sur les bords.
– Une forte compression (de l'ordre de 100) s'effectue dans la rétine passant de 10^8 photorécepteurs à 10^6 axons dans le nerf optique. En imaginant une activité des cellules ganglionnaires comprise entre 1Hz et 1 kHz, cela engendre une activité globale au niveau du nerf optique de 1 MHz à 1GHz, fréquence qui peut être atteinte avec le protocole de communication parallèle AER.

1.3.2 Les Rétines artificielles

Contrairement au capteur d'image traditionnel, les rétines neuromorphiques cherchent à recréer le fonctionnement de l'œil au plus proche de la nature. Les premières, la « Silicon Retina » (Mead and Mahowald (1988); Mahowald and Mead (1991)) et l' « Adpative Retina » (Mead (1989)), furent développées par Carver Mead, l'initiateur du neuromorphique. Ces rétines artificielles intègrent les cônes (photorécepteurs) ainsi que les cellules horizontales (réseau de résistance) et les cellules bipolaires. Ces implémentations diffèrent des capteurs d'images traditionnels de plusieurs façons. En premiers, ces rétines sont asynchrones (pas d'horloge, pas de trames) et utilisent, comme dans le cerveau, des signaux analogiques pour traiter et transporter l'information. Ensuite, elles sont capables de supprimer des informations redondantes (information qui n'évolue pas dans le temps) grâce aux cellules horizontales qui permettent de créer une moyenne spatiale et donc une adaptation. La réponse de la cellule bipolaire est seulement créée en cas de changement d'information au niveau du champ récepteur. Le fonctionnement de ces rétines est si proche de la rétine humaine qu'elles peuvent même être sujettes aux mêmes illusions d'optique (Mahowald and Mead (1991)). Par exemple, un carré gris sur fond noir apparait plus sombre que sur fond blanc.

La rétine DVS128 (Lichtsteiner et al. (2008)) est un exemple de rétines neuromorphiques à l'état de l'art. Cette rétine, de 128×128 pixels, est asynchrone et permet de ressortir deux types d'évènements pour chaque pixel, un évènement ON en cas d'augmentation de la luminosité et un évènement OFF en cas de diminution de la luminosité au niveau du pixel (figure 1.6). Le temps de réponse de chaque pixel est au minimum $15\mu s$ avec une illumination supérieure à 1 Klux. Lors de la création d'une information (ON ou OFF), la rétine retourne une adresse AER (Address Event Representation Boahen (1998, 2000); Mahowald (1994)) sur 15 bits (Abscisse x sur 7 bits, ordonnée y sur 7 bits, et la nature de l'évènement (ON ou OFF) sur 1 bit). Dans ce manuscrit, ces données AER correspondent aux évènements d'entrée des réseaux de neurones impulsionnels simulés et testés.

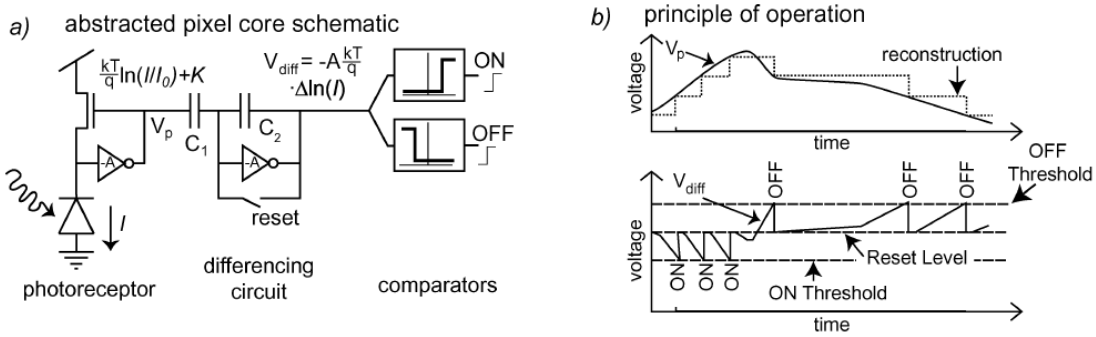


FIGURE 1.6 – Rétine AER - a) Schéma du pixel b) Principe d'opération d'un pixel de la rétine (Lichtsteiner et al. (2008)).

1.4 Le cortex visuel

1.4.1 Organisation du cortex

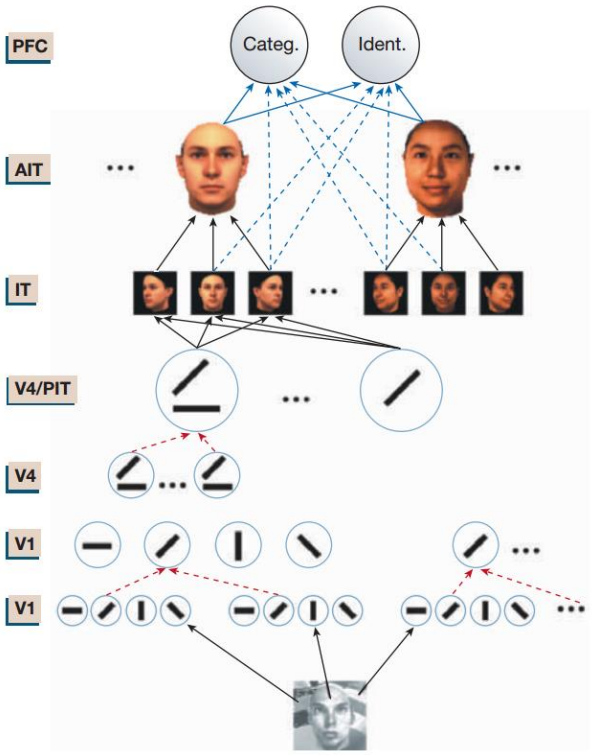


FIGURE 1.7 – Modèle simplifié, les ronds représentent les neurones et les flèches représentent leurs connexions. Les flèches pleines représentent un apprentissage de type généralisation et les flèches pointillées représentent des opérations de type max qui permettent une invariance de position et d'échelle.(Tomaso Poggio (2004))

Dans cette partie, nous allons voir comment les neurones sont organisés dans le cortex visuel pour reconstruire des informations de haut niveau et nous permettre de reconnaitre des motifs complexes. Le cortex visuel représente environ 20 % du cortex cérébral et contient environ 5 milliards de neurones (Wandell et al. (2007)). Il peut être divisé en plusieurs couches ($\approx$ 16, V1, V2, V3, V4, etc ..) en fonction de l'architecture, de la connectivité, de la topographie visuelle ou des caractéristiques fonctionnelles de ces couches (Van Essen (2003)), mais le sujet est toujours l'objet de débat (nombre de couches, délimitation). L'information circule principalement dans un seul sens (V1 -> V2 -> V3 . . .), ce qui permet un temps de réponse très rapide du cortex visuel (figure 1.8). Cela à été démontré par Thorpe dans le cortex visuel du singe ((figure 1.8), le temps de réaction du singe est en moyenne de 250 à 260 ms, mais peut atteindre un minimum de 180 ms. L'information perçue par la rétine atteint V1 en 50-60 ms et la reconnaissance complète d'éléments haut niveau est atteinte en 80-100 ms ce qui implique des connexions dans un sens unique. Toutefois on peut noter que des rétroactions des couches postérieures affectent aussi les couches inférieures (connexions excitatrices ou inhibitrices) pouvant influencer l'apprentissage des couches inférieures. La première couche, appelée aire primaire ou V1, est responsable de la détection de forts contrastes et peut-être assimilée à des filtres spatiotemporels de type transformés de Fournier ou filtre de Gabor. La couche V2 peut extraire des informations plus complexes d'orientation, de couleurs ou de mouvements. Sans entrer dans le détail des couches supérieures, on peut donc remarquer que l'information est disséquée dans (ou dès) les

premiers niveaux pour être reconstruite dans les niveaux supérieurs (figure 1.7).

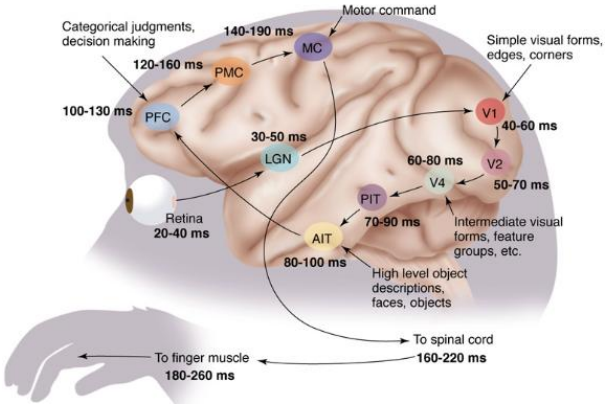


FIGURE 1.8 – Vitesse de perception visuelle et temps de réaction chez le singe. Le temps de réaction du singe est en moyenne de 250 à 260 ms, mais peut atteindre un minimum de 180 ms. L'information perçue par la rétine atteint V1 en 50-60 ms et la reconnaissance complète d'éléments haut niveau est atteinte en 80-100 ms. (Thorpe and Fabre-Thorpe (2001); Thorpe et al. (1996)

1.4.2 Modèle de cortex, le modèle HMAX

La plupart des architectures de reconnaissance sont basées sur le modèle du cortex visuel tiré des découvertes en neuroscience de Hubel et Wiesel (Hubel et al. (2009)). Ils décrivent un cortex primaire V1 composé de cellules *simples* et de cellules *complexes*. Dans un premier temps, les cellules simples ont pour rôle d'extraire des caractéristiques primaires de l'image perçue. Puis, dans un second temps, les cellules complexes rassemblent ces caractéristiques dans un voisinage spatial. Ce regroupement spatial, ou « pooling », permet d'obtenir une invariance en translation et en échelle de ces caractéristiques.

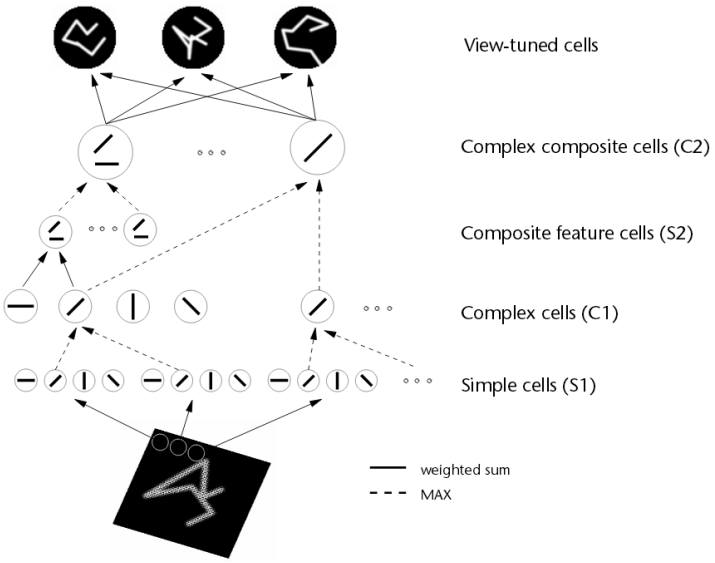


FIGURE 1.9 – Représentation du modèle HMAX. (Riesenhuber and Poggio (1999))

Le modèle HMAX, proposé par Riesenhuber et Poggio (Riesenhuber and Poggio (1999)) et étendu par Serre et al. (Serre et al. (2007)) est une succession de couches de cellules simples et de couches de cellules complexes. Ce modèle diffère notamment

de ces prédécesseurs par les cellules complexes. La fonction de regroupement pouvant être implémentée par une fonction linaire de sommation est ici implémentée par une fonction non-linéaire MAX.

Dans ce modèle, illustré figure 1.9, la première couche S1 est composée de cellules simples sélectives à l'une des quatre orientations (filtres de Gabor). Dans la seconde couche C1, chacune des cellules complexes reçoit les sorties (dans un voisinage) d'un groupe de cellules simples de la première couche répondant à une même orientation, mais à des positions ou des échelles légèrement différentes. Les cellules complexes réalisent une opération MAX sur ces entrées. Cela permet d'introduire une tolérance en translation et en échelle pour une même caractéristique primaire (orientation) (figure 1.10). Ensuite, une nouvelle couche de cellules simples (S2) rassemble les sorties des cellules complexes de la seconde couche pour devenir sélective à des caractéristiques plus complexes, par exemple la combinaison de deux orientations.

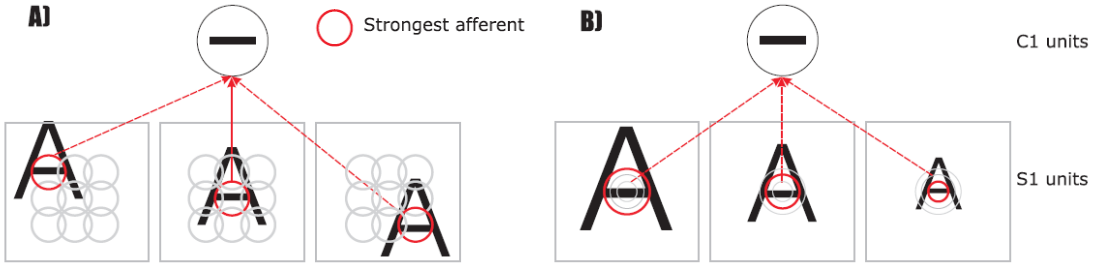


FIGURE 1.10 – Les cellules complexes (C1) assurent une invariance A) en translation et B) d'échelle des caractéristiques spécifiques des cellules simples (S1). (Serre and Riesenhuber (2004))

De ces travaux ont découlé plusieurs modèles d'architectures, le Neocognitron (Fukushima (1980)) ou les modèles de réseaux convolutionnels par exemple LeNet (LeCun et al. (1998)). On peut noter que d'autres modèles utilisent des mécanismes similaires (HOG (Histograms of Oriented Gradients, Dalal and Triggs (2005)), SIFT (Scale-Invariant Feature Transform, Lowe (1999)).

1.4.3 Les réseaux convolutionnels

Les convolutions spatiales permettent une extraction des caractéristiques primaires de l'image. Ces caractéristiques sont implémentées par les pondérations des masques de convolutions. Le résultat d'une convolution est alors proportionnel à la présence de cette caractéristique à cette localisation. Les convolutions permettent donc de localiser des caractéristiques dans l'image. Une couche de ces convolueurs peut donc s'apparenter à une couche de cellule simple du modèle HMAX, extractrice de caractéristiques. En utilisant des masques de convolution représentant une même caractéristique, mais pivoter, ou d'échelles différentes, cela permet avec une couche de cellule complexe de type MAX, d'implémenter une invariance en rotation, en échelle et en translation d'une même caractéristique. On peut donc remarquer que les réseaux convolutionnels convergent vers l'organisation du modèle HMAX.

Un exemple de réseaux convolutionnels est présenté figure 1.11. Cette architecture, destinée à la classification de 1000 classes d'objets, possède cinq couches de convolutions pour l'extraction de caractéristiques suivies de 3 couches complètement connectées. La première couche utilise 96 noyaux de convolutions de taille $11 \times 11 \times 3$(RGB) (figure 1.12), la seconde utilise 256 noyaux de taille $5 \times 5 \times 48$, la troisième utilise 384 noyaux de tailles $3 \times 3 \times 256$, la quatrième utilise 384 noyaux de taille $3 \times 3 \times 192$ et la cinquième couche utilise 256 noyaux de taille $3 \times 3 \times 192$. Cette architecture fut classée première

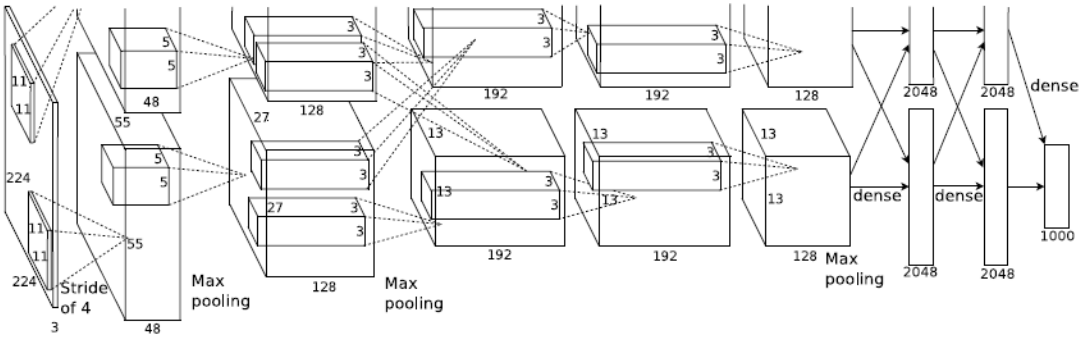


FIGURE 1.11

en 2012 pour le benchmark ImageNet (Krizhevsky et al. (2012)).

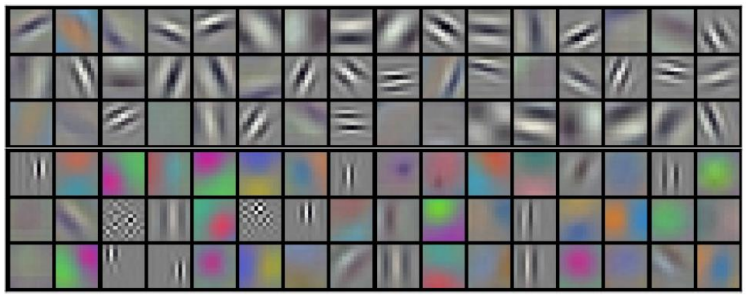


FIGURE 1.12 – 96 noyaux de convolutions de la première couche de convolution de l'architecture présentée figure 1.11 après un apprentissage supervisé (rétro-propagation du gradient). Ces noyaux correspondent aux caractéristiques primaires extraites de l'image d'entrée de $224 \times 224 \times 3$ pixels. On y retrouve des directions de plusieurs orientations et échelles.

Dans cette architecture, le convolueur est assimilé à un neurone et les valeurs du noyau aux pondérations synaptiques. Cet exemple montre qu'il n'est pas forcement nécessaire d'implémenter de grands champs synaptiques. En effet on peut remarquer que les noyaux de convolution ne sont pas de taille très importante (au maximum 11×11) ce qui nous sera utile dans notre étude pour estimer les besoins en taille d'une mémoire synaptique.

Les noyaux de convolution peuvent résulter, comme dans notre exemple, d'un apprentissage supervisé (de type rétro-propagation du gradient) ou d'un apprentissage non-supervisé, tiré des découvertes en neuroscience, la STDP, que nous allons présenter dans la section suivante.

Performance de ces modèles

Ces modèles de cortex visuel ont montré leur efficacité pour traiter des problèmes de perception visuelle comme la détection et la reconnaissance. Pour comparer leur efficacité, il existe des benchmarks très reconnus dans la communauté et qui utilisent différentes bases de données d'images. Table 1.1 présente les six meilleurs taux d'er-

reurs achevés par la communauté sur le benchmark MNIST. Ce benchmark de 10 classes (chiffres 0 à 9) est composé de deux bases de données d'images de chiffres manuscrits (28×28 pixels). La première, composée de 60 000 images, est dédiée à la phase d'apprentissage et la seconde, composée de 10 000 images, est dédiée à la phase de test. On y retrouve 4 architectures de type réseau de neurones convolutionnels, directement dérivés du modèle HMAX ce qui prouve l'efficacité de ces modèles de cortex. Cette efficacité est confortée par les résultats d'autres benchmarks, par exemple CIFAR-10 (10 classes (bateau, voiture, chat, grenouille, etc.), 50 000 images d'apprentissage et 10 000 images de test (32×32 pixels)) ou CIFAR-100 (20 superclasses, 100 classes, 500+100 images (32×32 pixels) par classe). (Benenson)

TABLE 1.1 – Meilleurs résultats du benchmark MNIST

Architecture	Taux d'erreur (%)	Référence
CNN	0.40	Simard et al. (2003)
CNN	0.39	Poultney et al. (2006)
MLP	0.35	Cireşan et al. (2010)
CNN committee	0.27	Ciresan et al. (2011)
MCDNN	0.23	Schmidhuber (2012)
DropConnect	0.21	Wan et al. (2013)

1.5 Méthode d'apprentissage

1.5.1 STDP biologique

L'Homme a depuis longtemps remarqué que le cerveau apprenait plus facilement les séquences qui se répètent dans le temps (Markram et al. (2011); Pavlov and Anrep (1927)). Cette observation fut plus tard précisée par le neuropsychologue Donald Hebb (Hebb (1949)), qui émit ce postulat : "Let us assume that the persistence or repetition of a reverberatory activity (or "trace") tends to induce lasting cellular changes that add to its stability.[...] When an axon of cell A is near enough to excite a cell B and repeatedly or persistently takes part in firing it, some growth process or metabolic change takes place in one or both cells such that A's efficiency, as one of the cells firing B, is increased". La neurobiologiste Carla Shatz synthétisa plus tard ce postulat avec « cells that fire together wire together » (les cellules qui s'activent ensemble se connectent ensemble) (Schatz (1992)).

La plasticité synaptique ou Spike Timing Dependant Plasticity (STDP) permet la modification des poids synaptiques en fonction de l'activité des neurones par le biais de mécanismes de potentialisation à long terme (LTP) (mis en évidence par Lømo (1966, 2003)) ou de dépression à long terme (LTD) (mis en évidence par Ito and Kano (1982)). La plasticité synaptique joue un rôle important dans les mécanismes d'apprentissage du cerveau. Ce procédé suit la règle de Hebb, mais y ajoute une précision temporelle. Une potentialisation ou une dépression a lieu lors de corrélation temporelle entre une impulsion pré-synaptique (impulsion du neurone d'entrée) et une impulsion post-synaptique (impulsion du neurone de sortie) et plus précisément, en fonction du temps relatif $\Delta t = t_{post} - t_{pré}$ (de l'ordre de la milliseconde) entre l'impulsion pré-synaptique et post-synaptique. Les tests in-vivo de Bi et Poo (Bi and Poo (1998)) démontrent une forme de STDP biologique (figure 1.13). Quand une impulsion pré-synaptique précède une impulsion post-synaptique (Δt positif), alors la synapse réalise une LTP. Dans

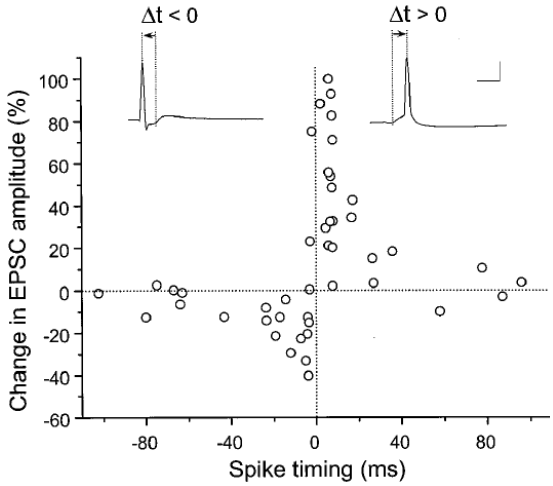


FIGURE 1.13 – Fenêtre temporelle STDP relevé par Bi et Poo (Bi and Poo (1998))

l'autre cas, quand une impulsion post-synaptique précède une impulsion pré-synaptique (Δt négatif), alors la synapse réalise une LTD. Plusieurs formes de STDP ont été observées dans la nature (figure 1.14). Ces modifications du poids synaptique, qui compose la mémoire à long terme, sont majoritairement non-volatiles et peuvent durer entre plusieurs heures et plusieurs mois (Caporale and Dan (2008)).

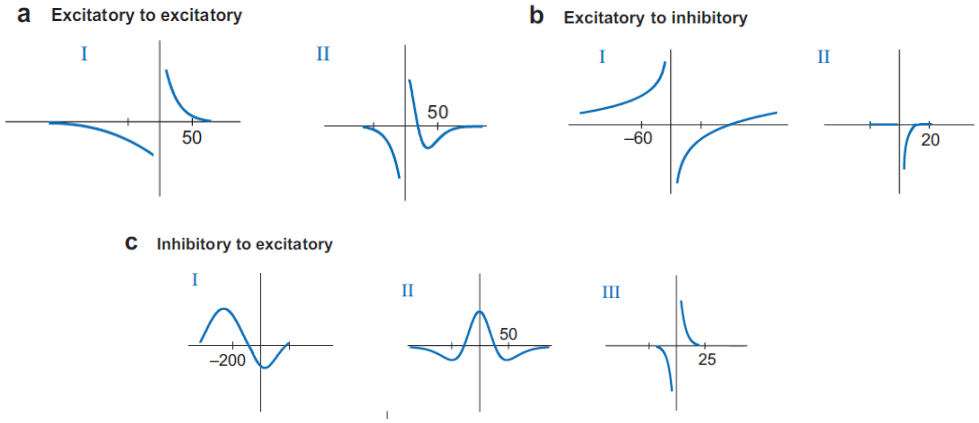


FIGURE 1.14 – Différente fenêtre temporelle de STDP. A) fenêtre temporelle des connexions excitatrices vers excitatrices. B) fenêtre temporelle des connexions excitatrices vers inhibitrices. A) fenêtre temporelle des connexions inhibitrices vers excitatrices. (Caporale and Dan (2008))

1.5.2 Apprentissage non-supervisé avec STDP

En 2007, l'équipe de Simon Thorpe a publié les premiers travaux mettant en place un mécanisme d'apprentissage STDP (Masquelier and Thorpe (2007)). Avec une architecture 5 couches de type HMAX (S1-C1-S2-C2-Classificateur), il a pu démontrer un apprentissage non-supervisé au niveau de la couche S2 sur une base de données comprenant des visages et des motos. Il utilise un réseau de neurones impulsionnels basé sur un codage « time-to-first-spike » (Temps de la première activation). Avec ce codage, le neurone qui s'active en premier est le neurone dont la réponse est la plus forte.

La compétition entre les neurones durant l'apprentissage est réalisée grâce à la période d'inhibition latérale. Cela permet à un neurone de devenir sélectif à un pattern

d'entrée et d'éviter que d'autres neurones apprennent ce même pattern. Lors de l'activation d'un neurone, et donc lors de l'apprentissage du pattern responsable de son activation, ce neurone désactive l'ensemble des autres neurones (le plus souvent, ceux présents sur la même couche que le neurone activé) pendant une période T_{INHIB}. C'est pourquoi ce type de réseau est décrit comme « winner take all ». La durée de cette période d'inhibition dépend principalement de la dynamique des stimuli d'entrée. Dans cette architecture, une inhibition latérale locale est présente dans chaque couche.

Les cellules S1 réalisent des convolutions spatiales 5×5 dont les noyaux correspondent à des directions de type filtre de Gabor (quatre orientations $\frac{\pi}{8}$, $\frac{\pi}{4} + \frac{\pi}{8}$, $\frac{\pi}{2} + \frac{\pi}{8}$,$\frac{3\pi}{4} + \frac{\pi}{8}$) à cinq différentes échelles (100%, 71%, 50%, 35%, et 25%). Cette première couche possède donc $4 \times 5 = 20$ S1 maps. Le temps d'activation de chaque neurone est inversement proportionnel à la valeur de la convolution spatiale (plus la caractéristique est présente à cet endroit, plus le neurone s"active tôt).

Les cellules C1 propagent la sortie la plus active de la couche S1 dans un voisinage de 7×7 cellules S1 d'une seule map (c'est-à-dire une certaine orientation à une certaine échelle). La couche C1 effectue donc un sous-échantillonnage ainsi qu'une invariance en translation.

Les cellules S2 représentent un niveau intermédiaire de complexité. En effet, grâce à l'apprentissage STDP, elles permettent d'apprendre une combinaison d'orientation pour créer des caractéristiques plus complexes. Chaque cellule S2 reçoit les sorties de 16×16 cellules C1 des quatre orientations à une certaine échelle (soit 96×96 entrées).

La règle STDP mise en œuvre est dérivée des découvertes en neuroscience. En fonction des temps d'arrivés des impulsions pré et post-synaptique ($T_{pré}$ et T_{post}), la synapse effectue une LTP si $T_{pré} - T_{post} < 0$ et augmente son poids de ΔW^+ (équation 1.2a), ou effectue une LTD si $T_{pré} - T_{post} > 0$ et diminue son poids de ΔW^- (équation 1.2a).

$$\Delta W^+ = a^+.W(1-W) \text{ si } T_{pré} - T_{post} \leq 0 \tag{1.2a}$$
$$\Delta W^- = a^-.W(1-W) \text{ si } T_{pré} - T_{post} > 0 \tag{1.2b}$$

Nous allons maintenant nous intéresser à l'apprentissage effectué sur cette couche S2. La figure 1.15 montre l'évolution de la reconstruction des visages et des motos due à l'apprentissage STDP en fonction du nombre d'activations post-synaptique. On peut voir que la règle STDP permet une extraction des caractéristiques les plus fortes de l'image.

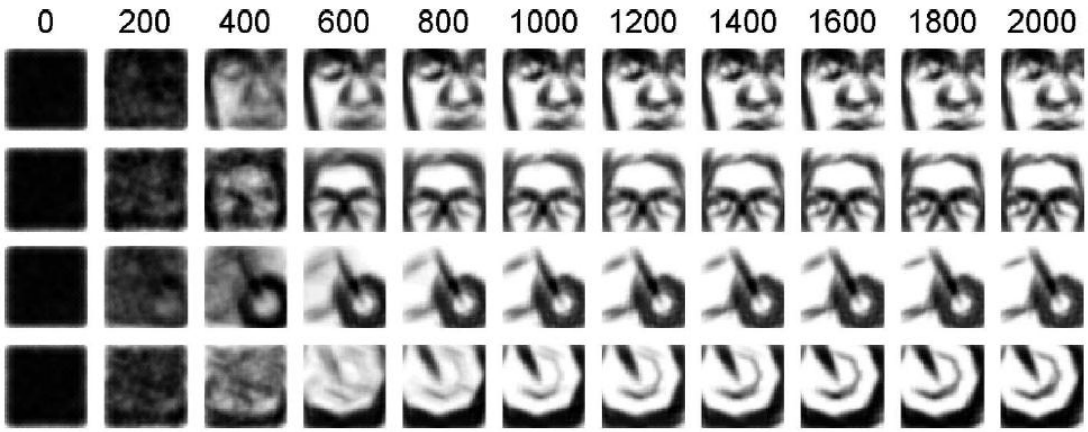


FIGURE 1.15 – l'évolution de la reconstruction des visages et des motos due à l'apprentissage STDP en fonction du nombre d'activations post-synaptique. (Masquelier and Thorpe (2007))

1.5.3 STDP simplifiée

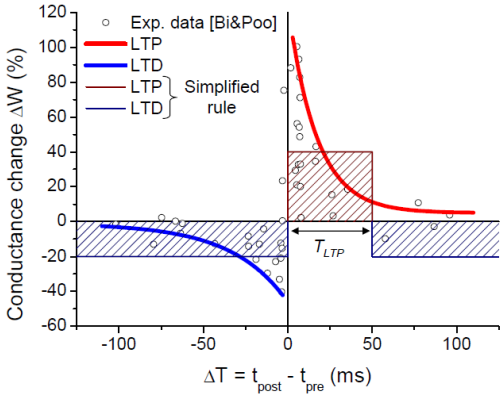


FIGURE 1.16 – Fenêtre temporelle de la méthode d'apprentissage STDP. Seules les synapses ayant entrainées l'activation du neurone dans la fenêtre temporelle T_{LTP} réalisent une LTP et voient leur poids synaptique augmenter de la valeur $\Delta W+$, toutes les autres réalisent une LTD et voient leur poids synaptique diminuer de la valeur $\Delta W-$

Nous avons vu qu'il existe plusieurs types de fenêtres temporelles pour la STDP. Dans nos travaux, nous utilisons une règle STDP simplifiée dont la fenêtre temporelle est présentée figure 1.16 et introduite par Bichler et al. (2012a). Cette règle diffère des résultats de mesures in-vivo de Bi et Poo (Bi and Poo (1998)), principalement en deux points.

$$W = W + \Delta W^{+} \text{ si } T_{pré} - T_{post} \leq T_{LTP} \tag{1.3a}$$

$$W = W + \Delta W^{\text{ si }} T_{pré} - T_{post} > T_{LTP} \tag{1.3b}$$

- Dépression à long terme (LTD) généralisé. En effet, une LTD est réalisée sur toutes les synapses qui ne se sont pas activées avant une impulsion post-synaptique. En résumé, toutes les synapses qui n'ont pas contribué à l'activation post-synaptique voient leur poids synaptique décrémenté (ΔW^{-}) (équation 1.3b). Dans le cas contraire, les synapses ayant contribué à l'activation du neurone post-synaptique, en s'activant pendant la fenêtre temporelle T_LTP, réalisent une LTP et voient leur poids synaptique incrémenté (ΔW^{+})(équation 1.3a). Cette technique d'apprentissage s'effectue lors de chaque activation neuronale. Cette modification permet de diminuer l'ensemble des poids synaptique dont les entrées n'ont pas contribué à l'activation du neurone.
- Discrétisation du ΔW. Les résultats de Bi et Poo, figure 1.13, montrent que la modification de l'amplitude du poids synaptique lors d'une LTP ou d'une LTD peut prendre différentes valeurs en fonction de la valeur $\Delta T = T_{pré} - T_{post}$. Dans notre modèle simplifier, seules deux valeurs sont possible, ΔW^{+} en cas de LTP et ΔW^{-} en cas de LTD (figure 1.16).

1.6 Systèmes neuromorphiques dédiés à la vision

1.6.1 Première génération et approche classique

Les premiers systèmes neuromorphiques dédiés à la vision sont basés sur les premières et secondes générations de réseaux de neurones. Ces systèmes, dont quelque un sont présentés figure 1.17, ont connus leurs apogées dans les années 1990. Bien que ces architectures soient plus efficaces (rapidité, consommation) que les architectures traditionnelles de type Von Neumann, le développement très rapide du CMOS (loi de Moore)

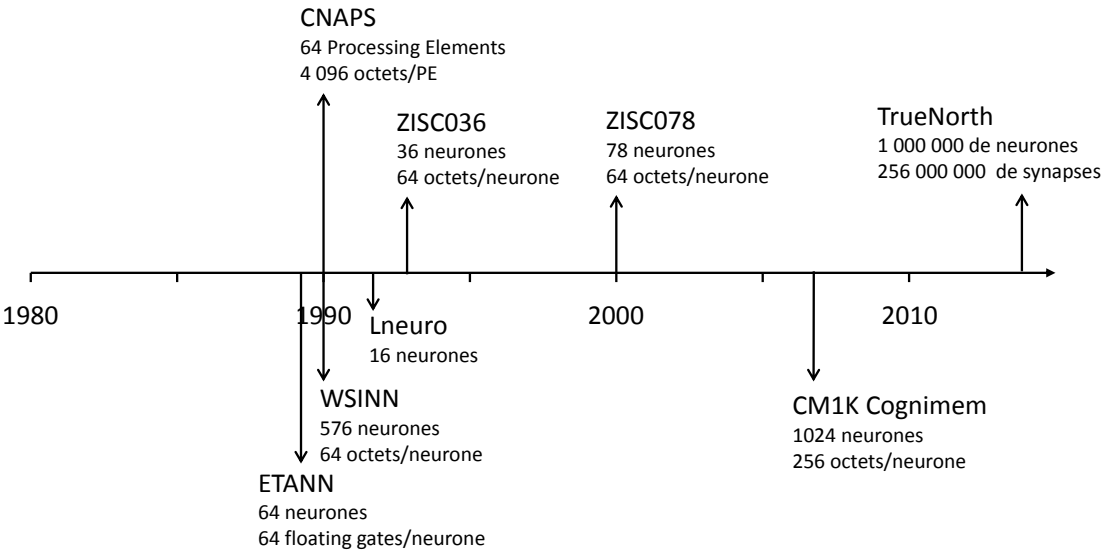


FIGURE 1.17 – Historique des implémentations matérielles de réseaux de neurones de première et seconde génération. ETANN, 1989 (Holler et al. (1989)). CNAPS, 1990 (Hammerstrom (1990)). WSINN, 1990 (Yasunaga et al. (1990)). Lneuro, 1992 (Mauduit et al. (1992)). ZISC036, 1993 (David et al. (1999)). ZISC078, 2000. CM1K Cognimem,2006 (Inc (2014)). TrueNorth, 2014 (Merolla et al. (2014)

permettait aux architectures traditionnelles de rattraper leur retard en quelques années. La troisième génération de réseaux artificiels, les réseaux impulsionnels, sont directement tiré des découvertes en neuroscience. Ces réseaux permettent de mettre en place le codage temporel de l'information, l'implémentation de l'apprentissage non-supervisé STDP et permettent aussi une compatibilité avec les rétines artificielles impulsionnelles.

1.6.2 Réseaux impulsionnels

Il est maintenant intéressant d'étudier une implémentation matérielle d'un modèle d'architecture impulsionnel. Dans cette partie, nous allons décrire un système complet neuromorphique dédié à la vision : le projet CAVIAR (Context Aware Vision and Recognition) (Serrano-Gotarredona et al. (2009)). Le but de ce projet est la création d'un système complet réalisant l'ensemble des différentes étapes de perception de l'environnement visuel, suivi de l'analyse des informations, puis d'une prise de décisions et enfin d'une action. Ce projet, schématisé figure 1.18, a notamment permis la création de 5 puces différentes que nous allons maintenant décrire :

1. **Perception de l'information visuelle - La rétine AER** La rétine AER créée pour ce projet est la rétine artificielle présentée précédemment dans ce chapitre dans la section 1.3.2. Pour rappel, cette rétine de 128×128 pixels ressort, de façon asynchrone, l'adresse AER d'un pixel lors de la variation de son illumination ainsi que le type d'évènement (ON ou OFF). Le flux AER est alors transmis à la puce de convolution 2D.
2. **Traitement bas niveau - Puce de convolution** Cette puce permet d'effectuer une extraction des caractéristiques primaires de l' « image » à l'aide de convolutions spatiales. Elle ne peut traiter, au maximum, qu'une image de 32×32 pixels. Il faudrait implémenter $4 \times 4 \times 2$ puces pour couvrir les 128×128 pixels et les deux types d'évènements de la caméra AER en parallèle. Les auteurs ont plutôt choisi de sous-échantillonner l'information en entrée passant de 128×128 à 64×64 pixels et de fusionner les deux types d'évènements. Cela permet d'implémenter quatre puces sur un circuit imprimé et de couvrir l'ensemble du champ visuel

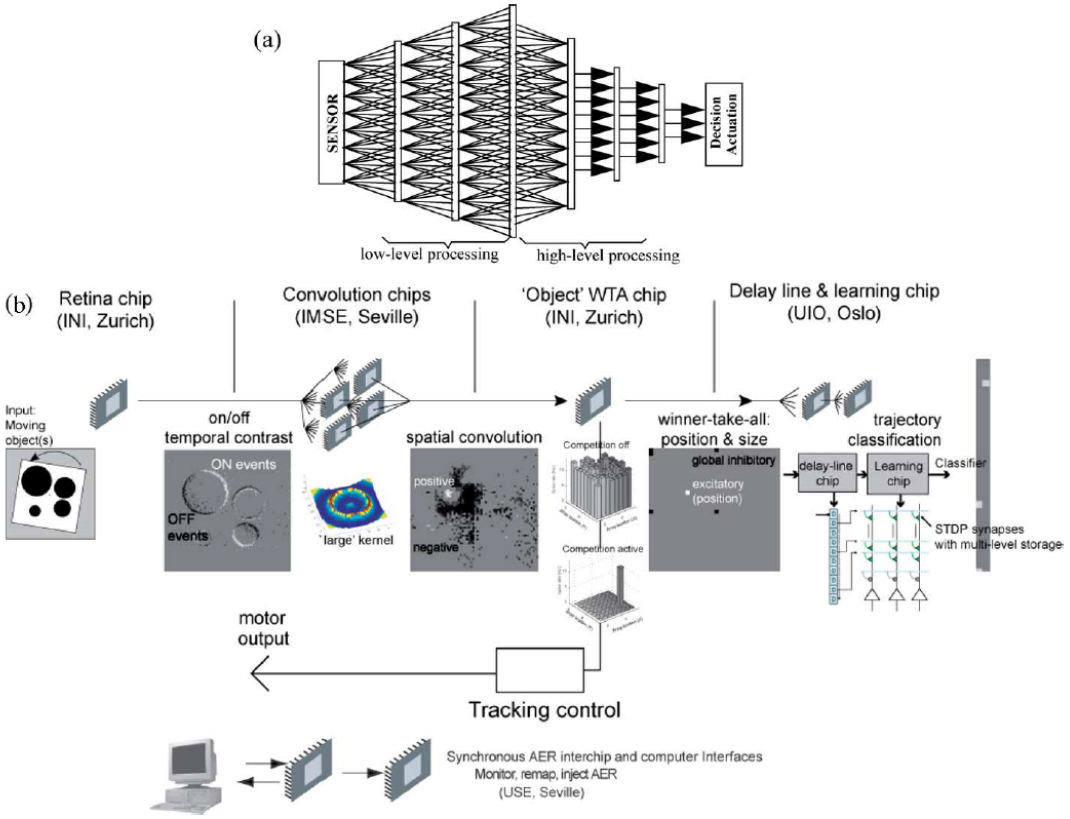


FIGURE 1.18 – Présentation du projet CAVIAR (Context Aware Vision and Recognition). (a) exemple d'un système complet (perception + traitement + action) bio inspiré réalisant les différentes fonctions de : 1) capter une image. 2) effectuer un prétraitement bas niveau de capture de caractéristiques (ex : convolution). 3) effectuer un traitement de compression d'information et un apprentissage haut niveau. 4) Faire un choix et envoyer une commande vers des actionneurs. (b) Composants du système CAVIAR avec une représentation de la sortie de chaque composant lors de la présentation d'un stimulus rotatif et la fonctionnalité basique de chaque composant est présentée sous chacun d'entre eux. (Serrano-Gotarredona et al. (2009))

perçu. Ces quatre puces peuvent recouvrir soit une même aire visuelle, soit une aire de 64×64 pixels. Les noyaux de convolution peuvent atteindre une taille de 31×31. Un exemple de noyau de détection de cercles de diamètres différents est présenté figure 1.19. Chaque sortie peut émettre deux types d'évènements (ON ou OFF) en fonction du seuil (positif ou négatif) dépassé.

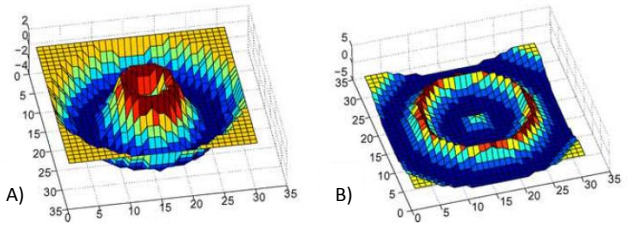


FIGURE 1.19 – Possible noyaux de la puce de convolution : Noyau de convolution permettant la détection d'un cercle ayant une circonférence A) proche de 4 pixels. B) proche de 9 pixels. (Serrano-Gotarredona et al. (2009))

3. **Traitement haut niveau – puce Winner-Take-All** Cette puce contient 32×32 ou 1024 neurones. Pour pouvoir traiter l'ensemble des données AER provenant de la puce de convolution, un second sous échantillonnage a été effectué en passant

de 64×64 à 32×32 (en fusionnant chaque groupe de pixels 2×2). Ces neurones sont découpés en quatre groupes de 16×16 ou 256 neurones dont 254 sont excitateurs et 2 sont inhibiteurs et chaque groupe traite une des quatre puces de convolution. Ces neurones implémentent un circuit Winner-Take-ALL, ce qui signifie que le neurone qui s'active en premier remporte la compétition et inactive alors l'ensemble des autres neurones. Elle permet dans un premier temps d'établir une compétition au sein d'une même puce de convolution pour déterminer la caractéristique la plus forte provenant de cette puce puis une seconde compétition pour déterminer la caractéristique la plus forte entre les quatre puces de convolution. Cette étape permet ainsi une réduction de l'information en ne conservant que l'information de localisation de la plus forte caractéristique. Cette information est ensuite sous-échantillonnée en un groupe de pixels 2×2, en fusionnant chacun des quatre groupes de pixels 16×16 et est envoyée en format AER 16 bits à la puce de retard.

4. **Dimension spatiale 2D vers 3D - Puce de retard** Cette puce permet d'ajouter une nouvelle dimension à l'information, le temps, grâce à un circuit de retard composé de 880 points d'accès sur 16 bits (adresse AER, nécessitant donc 16×880 soit 14 080 retardateurs (inverseurs) pour chaque ligne AER). Ce circuit permet simplement de recréer ce que nous faisons chaque jour dans notre esprit à savoir visionner un mouvement ou une trajectoire, en l'étalant dans le temps comme il est possible de le voir figure 1.20. Dans le démonstrateur CAVIAR, 3 points d'accès ont été choisis (0 s, 200 ms, et 400 ms) sur les 2×2 pixels d'entrées ce qui résultent en 12 ou $2 \times 2 \times 3$ pixels de sortie pour la prochaine puce, la puce d'apprentissage.

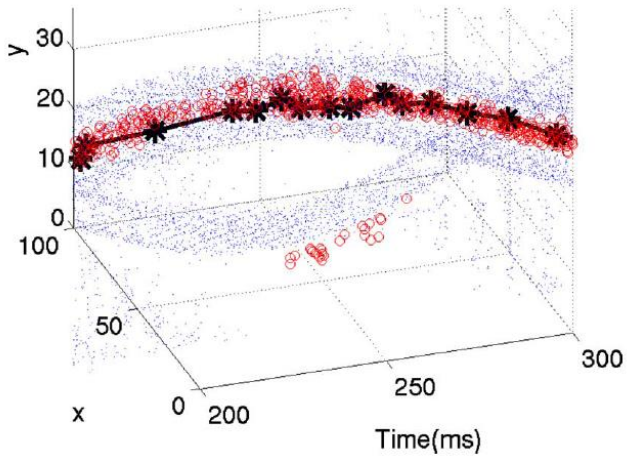


FIGURE 1.20 – Reconstruction 3D des évènements recueillis pendant une durée de 100 ms avec un stimulus correspondant à deux cercles de diamètre différent en rotation. Les points bleus correspondent aux évènements en sortie de la caméra AER, les cercles rouges correspondent à la sortie des puces de convolution et les étoiles noires correspondent à la sortie du circuit Winner-Take-All. (Serrano-Gotarredona et al. (2009))

5. **Puce d'apprentissage** Cette puce est composée de 32 neurones impulsionnels possédant chacun 64 synapses implémentant la règle d'apprentissage STDP proche de celle décrite dans ce chapitre dans la section 1.5.3. Les synapses sont implémentées par la mémoire classique à six niveaux. Chacun des neurones possède également une synapse inhibitrice connectée à la sortie de l'ensemble des autres neurones afin d'implémenter l'inhibition latérale WTA et permettre à un neurone d'être sélectif à un pattern en évitant que d'autres neurones apprennent

ce même pattern. La sortie de cette puce est donc une représentation haut niveau et compressée des stimuli d'entrées. Elle permet une classification de 32 catégories (32 neurones).

Expérimentation Avant apprentissage, lors de la présentation d'un stimulus rotatif représentant deux cercles de diamètre différent ainsi que deux autres formes, les 32 neurones de la puce d'apprentissage s'activent peu, excepté pour deux qui s'activent fortement, mais qui ne permettent pas de localiser l'emplacement du stimulus (figure 1.21 A)). Cela est dû à l'initialisation aléatoire des poids synaptiques. Après apprentissage (figure 1.21 B)), entre 7 et 9 neurones se sont spécialisés dans une des phases de rotation du stimulus. Il est maintenant possible de suivre l'emplacement du cercle en fonction des neurones de sortie.

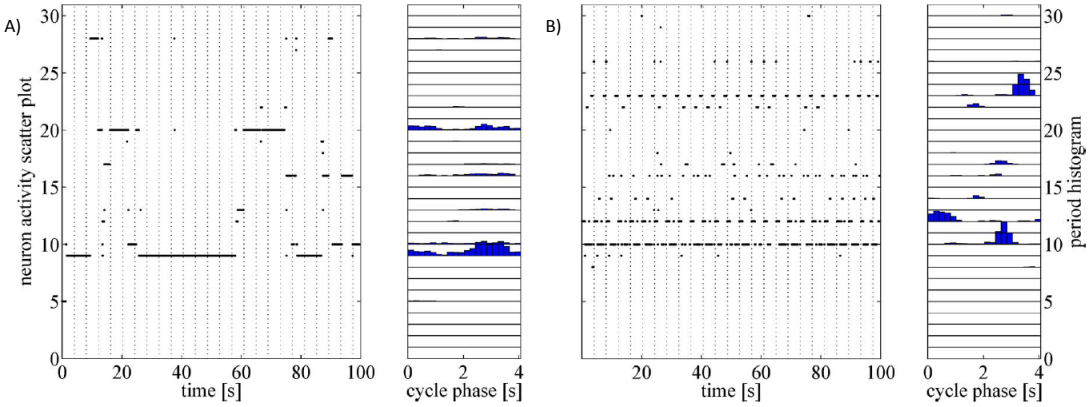


FIGURE 1.21 – Activité des neurones de la puce d'apprentissage A) avant B) après apprentissage. On peut voir qu'après l'apprentissage entre 7 et 9 neurones se sont spécialisés dans une phase de rotation d'un des cercles. (Serrano-Gotarredona et al. (2009))

Discussion L'architecture du projet CAVIAR permet de réaliser l'ensemble des étapes de perception, analyse et action et est entièrement événementielle, codée au format AER. On peut remarquer que cette architecture suit parfaitement les modèles du vivant décrits précédemment. En effet, dans un premier temps, la puce de convolution correspond à une couche de cellules simples (du modèle de Hubel et Wiesel (Hubel et al. (2009)), section 1.4.2) et permet l'extraction des caractéristiques primaires de l'environnement. Dans un second temps, la puce WTA implémente la couche de cellules complexes et réalise une opération HMAX des caractéristiques détectées dans la couche précédente.

1.7 Discussion et perspectives

De ce premier chapitre d'état de l'art, nous pouvons tout d'abord retenir que l'unité fonctionnelle du cerveau, le neurone, permet de traiter efficacement l'information. Un mécanisme de plasticité synaptique lui permet de devenir sélectif à un pattern répétitif. Ensuite, l'œil, organe de perception, permet de supprimer les informations redondantes et qu'il effectue une première compression des informations transmises au cortex visuel. Enfin, l'organisation des neurones dans le cortex visuel nous apprend que l'information, fractionnée dans ses formes les plus simples dans les couches primaires est ensuite regroupée au fil des couches pour reformer des formes de plus grande complexité.

Nous nous sommes ensuite intéressés aux implémentations neuromorphiques de ces différentes structures. En premier, nous avons vu qu'à l'instar des premières et secondes

générations, les neurones impulsionnels reproduisent le comportement des neurones biologiques et permettent l'implémentation du mécanisme d'apprentissage STDP, calqué sur la biologie. Ensuite, une rétine impulsionnelle permet d'imiter le comportement de l'œil en effectuant une discrétisation asynchrone de l'information à partir du pixel. Cette information impulsionnelle, non redondante, est codée au format AER ce qui la rend compatible avec les neurones impulsionnels. Enfin, des modèles de cortex visuel de type HMAX ou CNN schématisent les manières d'architecturer ces neurones pour leur permettre de devenir sélectifs à des formes complexes ce qui a été démontré expérimentalement dans le projet CAVIAR.

Dans la suite de ce manuscrit, nous allons focaliser notre étude sur les neurones impulsionnels et la règle d'apprentissage STDP, et notamment leurs optimisations dans le but d'améliorer leur intégration et les rendre compatibles au monde de l'embarquée.

Chapitre 2

Optimisation d'un réseau de neurones impulsionnels

Sommaire

2.1 Introduction 27
2.2 Application Détection de véhicule 29
2.3 Simulateur XNET 30
2.4 Impact de la résolution des poids synaptiques. 31
2.5 Impact d'une décroissance linéaire du potentiel membranaire 34
2.6 Impact du remplacement des fenêtres temporelles par une FIFO 36
2.7 Résultat comprenant l'ensemble des modifications 38
2.8 Mises en place des optimisations 39
2.9 Discussion et perspectives 40

2.1 Introduction

Les réseaux de neurones impulsionnels artificiels peuvent être implémentés avec différents degrés de complexités. Celle-ci dépend du compromis entre la fidélité aux modèles biologiques recherchée, l'efficacité en terme d'application et de l'adéquation des calculs à effectuer aux technologies envisagées pour les mettre en œuvre. Dans un contexte embarqué, l'optimisation de ces réseaux a pour but un gain en superficie et en énergie tout en conservant sa fonctionnalité et sa capacité d'apprentissage. Il est donc indispensable d'analyser ce réseau dans son ensemble pour y faire apparaître les différentes composantes. Un réseau de neurones impulsionnels est composé de trois parties majeures : le neurone impulsionnel, la synapse et une technique d'apprentissage (figure 2.1).

Dans ce chapitre, nous nous intéressons à l'étude de chacune de ces parties afin de diminuer leur surface CMOS et leur consommation d'énergie dans le cadre d'une implémentation purement numérique. Pour valider les modifications apportées à ce réseau, adapté à la détection à partir d'une rétine AER asynchrone, et vérifier que son apprentissage et sa fonctionnalité restent inchangés, nous nous sommes appuyé sur une application développée dans notre laboratoire : la détection de véhicules (Bichler et al. (2011)).

Dans un premier temps, nous allons présenter l'application de détection de véhicule ainsi que l'outil de simulation XNET (Bichler et al. (2013b)). Puis, nous exposerons les modifications apportées à chacune des trois parties du réseau de neurones impulsionnels. En premier, nous avons réalisé une étude sur l'influence de la résolution des poids

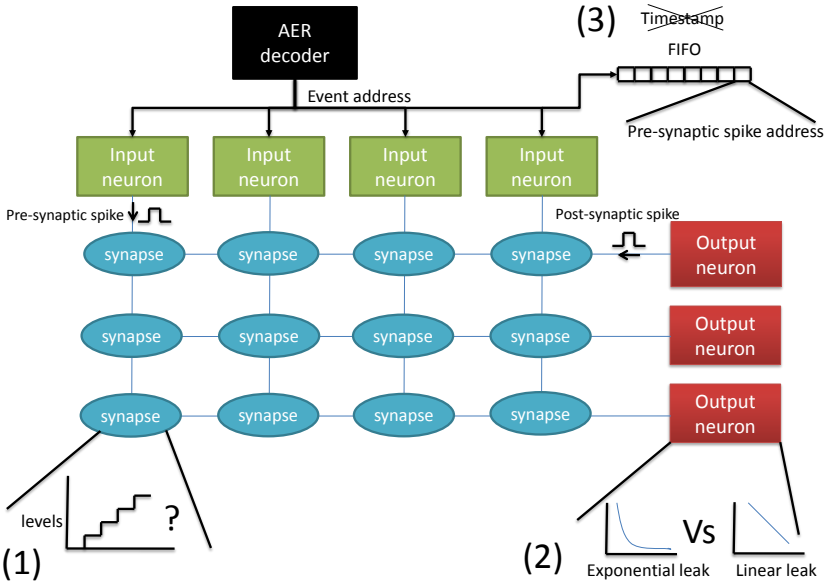


FIGURE 2.1 – Schéma d'un réseau de neurones impulsionnels. Les trois parties majeures sont étudiées dans ce chapitre, la synapse, le neurone et la technique d'apprentissage STDP. 1) Résolution optimale et minimale des poids synaptiques, 2) Simplification du calcul de la composante de fuite du neurone LIF, 3) Implémentation d'une STDP basée sur l'ordre d'arrivée des impulsions grâce à une FIFO.

synaptiques pour ce type d'application. Pour un gain de superficie et d'énergie, il est intéressant de minimiser la résolution du poids d'une synapse (en bit) et donc de minimiser la taille de la mémoire (et donc le nombre de dispositifs memristifs) requise pour implémenter la totalité des synapses (cette application requiert sous sa forme originale, par exemple, un nombre total de 2 000 000 synapses). En second, nous allons étudier le neurone impulsionnel LIF «*Leaky* Integrate and Fire » et plus particulièrement sa composante de fuite (leak). En effet, si la décroissance exponentielle du potentiel membranaire est bien adaptée à une implémentation par un réseau analogique (comme la décharge d'une capacité dans une résistance) dont le fonctionnement est régi par une équation différentielle du premier ordre, il ne l'est pas autant lorsqu'il s'agit de simuler cette décroissance par un calcul numérique. L'implémentation de la fuite exponentielle génère un coût CMOS non négligeable dans l'ALU du système dans la perspective d'une implémentation numérique. Cette étude porte sur la modification de la décroissance exponentielle par une décroissance plus simple à l'implémentation, une décroissance linéaire. Enfin, nous allons porter notre étude sur la méthode d'apprentissage STDP. Cette méthode, basée sur le temps d'arrivée des impulsions, implique un marquage temporel du dernier instant d'activation de chaque neurone pour pouvoir les comparer en implémentant des compteurs de temps. Ces compteurs ont une consommation d'énergie dynamique importante, car leur état s'incrémente à chaque coup d'horloge. Pour éliminer ces compteurs énergivores, nous allons modifier la méthode STDP en faisant l'hypothèse, que l'on validera, que l'ordre d'arrivée des impulsions est aussi important que leur temps d'arrivée. Cela nous permet d'implémenter une mémoire FIFO à la place des compteurs de temps, qui sera partagé par l'ensemble des neurones d'une même couche.

Ce chapitre présente les travaux publiés dans la conférence IJCNN (International Joint Conference on Neural Networks) 2013 (Roclin et al. (2013)).

2.2 Application Détection de véhicule

Dans le domaine de la vision, les réseaux de neurones impulsionnels peuvent aborder deux grands types de problématique, les applications de détection ou les applications de reconnaissance. Dans ces travaux, nous avons choisi de nous baser sur une simulation expérimentée dans notre laboratoire, une application de détection de véhicule sur une portion d'autoroute. Une caméra AER (Lichtsteiner et al. (2008)), placée sur un pont au-dessus de l'autoroute 210 à Pasadena, Californie, USA, a enregistré les véhicules circulant sur l'autoroute. Dans cette séquence, il y a un total de 207 voitures circulant sur 6 voies de circulation pendant une durée d'une minute et vingt-six secondes. Ce flux AER asynchrone (disponible en ligne : DVS128 Dynamic Vision Sensor Silicon Retina data (2011)) correspond aux signaux d'entrée de notre réseau impulsionnel. La résolution de la caméra AER est de 128 × 128 pixels et chaque pixel peut émettre deux types d'événements, un évènement positif quand la luminosité croît (ON), ou un évènement négatif, quand la luminosité décroît (OFF). Notre réseau possède donc 32 768 (128 × 128 × 2) entrées AER distinctes. Le réseau originel simulé (Bichler et al. (2011)) possède deux couches complètement connectées. La première de 60 neurones ou chaque neurone possède 32 768 synapses et la seconde de 10 neurones avec 60 synapses pour chaque neurone. Le réseau comprend ainsi un total de 1 966 680 synapses. Dans cette simulation, il est montré que la première couche apprend des trajectoires partielles alors que la seconde regroupe ces trajectoires pour en créer une complète. Dans nos travaux, nous allons conserver uniquement la première couche de 60 neurones (figure 2.2), car la seconde couche n'améliore pas significativement le taux de détection pour ce type d'application. Après apprentissage, chaque neurone devient spécialisé pour une voie de circulation spécifique. Un neurone de sortie émet une impulsion quand un véhicule passe sur sa voie de circulation. Au total, six neurones se sont spécialisés (mieux que les autres) à chacune des 6 voies de circulation. Dans les travaux suivants, nous identifions respectivement les meilleurs neurones de chaque voie de circulation : « voie 1 », « voie2 », . . . , « voie 6 ». L'activité totale enregistrée par ces six neurones représente le nombre de véhicules qui ont circulé sur ces six voies de circulation. Ce réseau est capable de détecter les véhicules d'une manière complètement non supervisée. Pour simuler ce réseau, nous avons utilisé un simulateur développé dans notre laboratoire XNET.

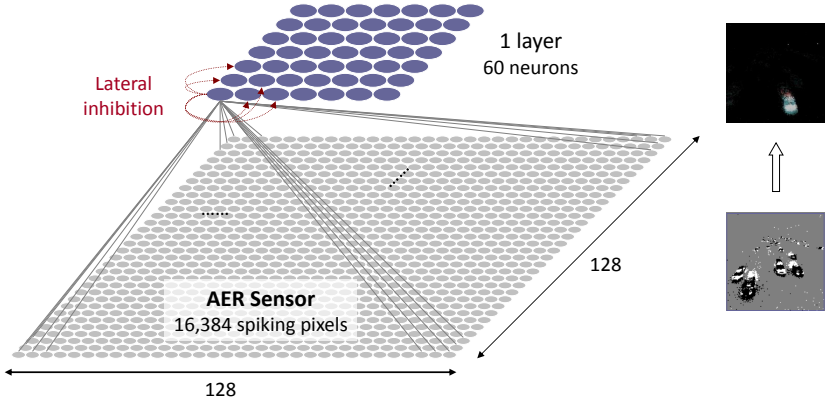


FIGURE 2.2 – Topologie du réseau implémentant la détection de véhicule sur une portion d'autoroute. Le flux AER en entré du réseau est généré par une camera AER possédant une résolution de 128 × 128 pixel où chaque pixel peut émettre deux types d'évènements, ON et OFF. Ce réseau est un réseau impulsionnel monocouche de 60 neurones. Après apprentissage, un neurone peut se spécialiser dans la détection de véhicule pour une voie de circulation particulière. Il y a 6 voies de circulation sur cette autoroute, pour un apprentissage réussi, six neurones doivent se spécialiser pour chacune d'entre elles.

2.3 Simulateur XNET

XNET est un simulateur évènementiel développé au CEA en C++ et dédié à l'étude de réseaux de neurones impulsionnels. Une description exhaustive de XNET est présentée dans la publication (Bichler et al. (2013b)). Cet outil permet d'explorer de nombreuses topologies de réseaux tout en nous permettant de modifier chaque structure du réseau indépendamment. Dans notre réseau de détection, six paramètres majeurs influent sur les neurones et les synapses, ces paramètres sont présentés dans le tableau 2.1. Cinq paramètres influent sur le neurone : son seuil, sa fenêtre de temps LTP, sa période réfractaire, sa période d'inhibition latérale et sa constante de temps de fuite. Pour les synapses, deux paramètres entrent en jeu : la valeur d'incrémentation et de décrémentation du poids d'une synapse lors de l'apprentissage et la résolution du poids synaptique.

TABLE 2.1 – Liste des paramètres des synapses et des neurones dans XNET

W_{len}	Résolution des poids synaptiques
I_{thres}	Seuil du neurone
T_{LTP}	Fenêtre de temps LTP
T_{refrac}	Période réfractaire
$T_{inhibit}$	Période d'inhibition
τ_{leak}	Constante de temps de fuite
$(\Delta w_+, \Delta w_-)$	Valeur de l'incrémentation ou de la décrémentation du poids synaptique lors de l'apprentissage

Pour évaluer l'impact des modifications sur la fonctionnalité et les performances d'un réseau, le simulateur XNET génère un score lors de chaque simulation évaluant sa capacité d'apprentissage. En effet, il est possible de donner un score à un réseau en fonction de l'activation des neurones de sortie. Pour l'application de détection de véhicule, un relevé à la main des temps de passage des véhicules sur chaque voie de circulation a été effectué. Ce relevé est ensuite comparé au train d'impulsion en sortie du réseau (voir Figure 2.3 a)). Pour comparer deux trains d'impulsion de signaux discrets, on calcule les produits de convolution entre chaque train d'impulsion et une gaussienne pour produire deux signaux continus (voir Figure 2.3 b)). L'erreur finale générée pour une voie de circulation est égale à l'intégration de la valeur absolue de la différence de ces deux signaux (voir Figure 2.3 c) et équation 2.1).

$$Error = \int |(Spk_{reference} * \phi) - (Spk_{neuron} * \phi)| \mathrm{d}t \tag{2.1}$$

Avec ce calcul, une erreur de 1 peut représenter soit un faux positif, le neurone émet une impulsion, mais il n'y a pas de voiture présente, soit un faux négatif, le neurone n'émet pas d'impulsion, mais une voiture est présente. Il serait possible d'intégrer séparément les valeurs positives et négatives de l'erreur afin de distinguer la non-détection de la fausse détection. Une erreur inférieure à 1 indique qu'une voiture a bien été détectée, mais que les deux impulsions ne sont pas exactement synchrones dans le temps. Dans l'exemple de la Figure 2.3, le neurone émet un faux positif à la 40e seconde et manque deux voitures à la 70e seconde de la séquence. L'erreur de cet exemple est de 3.36, il y a eu 3 erreurs et plusieurs évènements pas parfaitement alignés temporellement.

Pour trouver les paramètres optimaux d'un réseau, XNET permet d'effectuer des évolutions génétiques simples. Un grand nombre de simulations est effectué où chaque paramètre peut évoluer indépendamment des autres. Ces paramètres commencent à une valeur donnée puis évoluent aléatoirement selon une distribution normale (avec

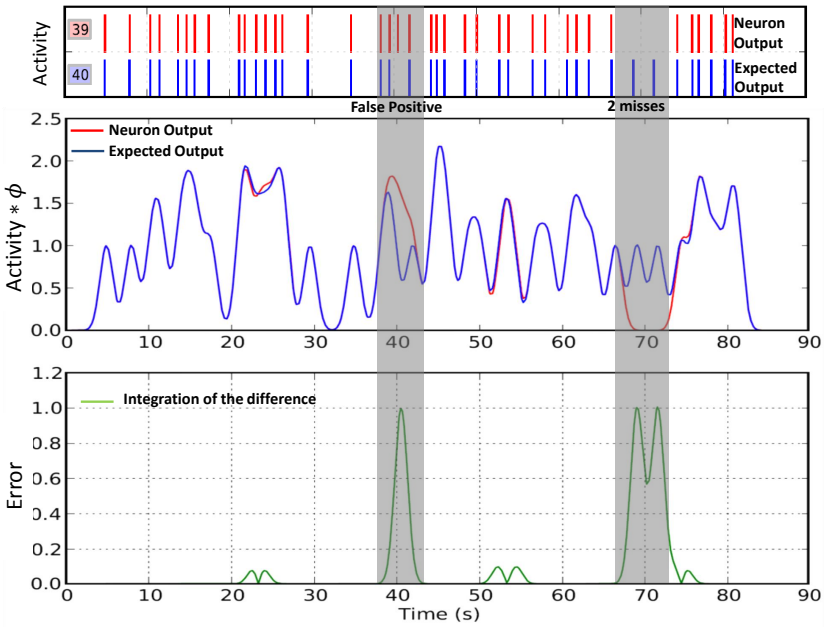


FIGURE 2.3 – Calcul du nombre d'erreurs - a) Comparaison des trains d'impulsions, en rouge l'activité d'un neurone de sortie et en bleu, l'activité attendue. b) Signaux continus des trains d'impulsions convolués avec une gaussienne. c) Valeur absolue de la différence des deux convolutions précédentes. – L'erreur finale est égale à l'intégration de cette différence. Dans cet exemple, le neurone émet un faux positif à la 40e seconde et manque deux voitures à la 70e seconde de la séquence. L'erreur pour ce neurone sur cette voie de circulation est de 3,36.

un écart type de 0,4 dans cette étude). Dans cette étude, nous avons accompli des évolutions génétiques de 30 générations avec 180 simulations pour chaque génération afin de trouver les meilleurs paramètres, ceux dont la simulation génère le plus petit score d'erreur. Après chaque génération, une sélection naturelle des meilleurs scores est effectuée. Nous procédons à une sélection des quatre meilleurs scores et la nouvelle génération utilise les paramètres de ces quatre vainqueurs comme valeurs moyennes pour la génération suivante. A chaque nouvelle génération, si quatre nouveaux vainqueurs sont déterminés alors l'écart type est multiplié par 1,5. Dans le cas contraire, si aucun nouveau vainqueur n'est déterminé, l'écart type est divisé par deux et les quatre derniers vainqueurs connus sont utilisés comme base.

Dans cette étude, nous avons modifié le réseau original point par point et effectué une évolution génétique après chaque modification. Nous étudions les résultats obtenus et si une évolution génétique produit un score inférieur à 10 alors nous estimons que la capacité d'apprentissage du réseau de neurones impulsionnels mis en œuvre est satisfaisante (un score de 10 pour un total de 207 voitures équivaut approximativement à un taux de reconnaissance de 95%).

2.4 Impact de la résolution des poids synaptiques.

Près de 2 millions de synapses sont nécessaires dans notre simulation. Leur implémentation requiert donc une étude approfondie surtout pour une application embarquée ou la taille et la consommation sont des facteurs primordiaux (Pfeil et al. (2012)). La résolution synaptique prend un sens aussi bien dans le monde analogique que numérique. Elle peut se traduire, dans un environnement analogique, par le nombre de niveaux de conductance d'un memristor ou, dans un environnement numérique, par le nombre de bits d'une mémoire classique par exemple de type FLASH ou SRAM. Cette étude

approfondie nous permettra de déterminer la résolution optimale et minimale afin de diminuer la taille de la mémoire synaptique tout en conservant une capacité de détection de véhicule maximale.

Afin de déterminer la résolution minimale qui n'altère pas significativement les performances, nous avons simulé l'application de détection de véhicule avec une résolution synaptique de 2 bits à 8 bits. Pour chaque résolution, nous avons effectué une évolution génétique afin de trouver les paramètres optimaux, ces paramètres sont présentés dans le Tableau 2.2. La moyenne des erreurs de chaque voie de circulation sur 100 simulations à chaque résolution est présentée dans la Figure 2.4.

TABLE 2.2 – Valeurs des paramètres neuronaux et synaptiques pour chaque résolution de poids synaptique (de 2 à 8 bits) de la simulation 2.4.

weight *(bits)*	I_{thres}	τ_{leak} *(ms)*	T_{LTP} *(ms)*	$T_{inhibit}$ *(ms)*	T_{refrac} *(ms)*	meilleur score
2	3539	386	13	212	388	6.68
3	3953	257	3	49	513	6.45
4	19 182	244	7	27	471	5.79
5	39 383	292	8	18	482	5.71
6	44 184	281	4	25	568	5.20
7	119 825	341	6	21	477	5.91
8	236 130	212	4	26	516	4.75

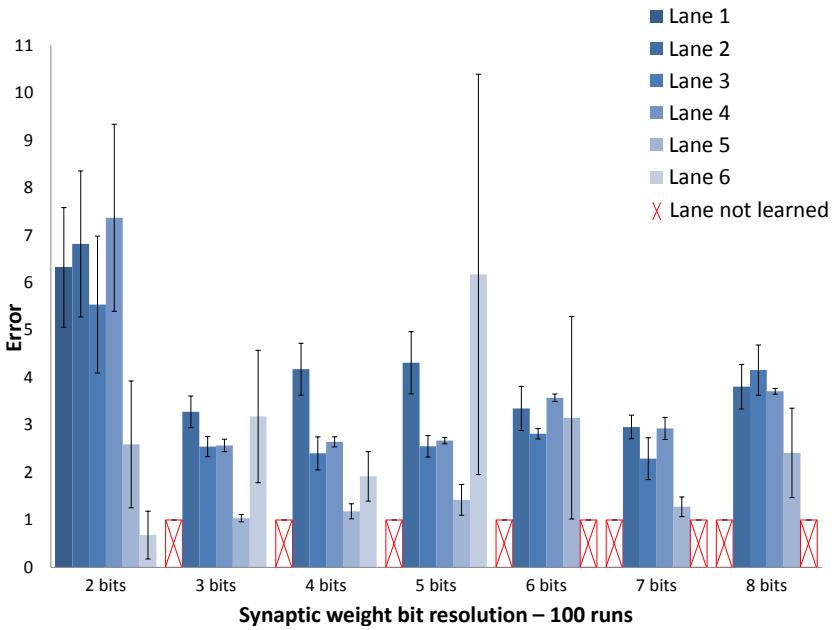


FIGURE 2.4 – Moyenne des erreurs pour chaque voie de circulation avec une résolution synaptique de 2 à 8 bits sur 100 simulations. Les voies avec une croix rouge sont les voies non apprises par le réseau.

Les résultats de la Figure 2.4 montrent une moyenne de 3 ou 4 erreurs par voie pour chacune des résolutions synaptiques excepté pour une résolution de 2 bits qui a une moyenne de 6 erreurs par voie. Cela signifie que le réseau est fonctionnel à partir d'une résolution de 3 bits. De ces résultats, on peut surtout remarquer que le seuil de neurone est le seul paramètre qui a vraiment évolué. Le seuil évolue, car il est dépendant de la valeur maximale que peut prendre le poids synaptique et donc, de la résolution synaptique. Les autres paramètres sont des paramètres temporels, qui eux ne sont pas

dépendants de la résolution des poids synaptiques, mais liés à une application donnée. En effet, la vitesse et la fréquence des voitures sont identiques entre chaque simulation.

Nous nous sommes alors demandé s'il était possible de trouver un jeu de paramètres valide qui fonctionnerait pour chaque résolution synaptique (de 2 à 8 bits) en normalisant le seuil I_{thres} en fonction du poids maximal $Wmax = (2^{W_{len}} - 1)$ (par exemple pour un poids synaptique de 3 bits, I_{thres} deviens $I_{thres} \times Wmax = I_{thres} \times 7$)). Nous avons effectué une évolution génétique exhaustive et le meilleur jeu de paramètres est présenté dans la table 2.3. La moyenne des erreurs sur 10 simulations avec ce jeu de paramètres est présentée dans la figure 2.5.

TABLE 2.3 – Valeurs des paramètres neuronaux et synaptiques de la simulation figure 2.5.

W_{len} *(bits)*	I_{thres} $(/W_{\max})$	τ_{leak} *(ms)*	T_{LTP} *(ms)*	$T_{inhibit}$ *(ms)*	T_{refrac} *(ms)*	$\Delta(w_+, w_-)$
2 to 8	1057	204	7.5	18	524	(3,1)

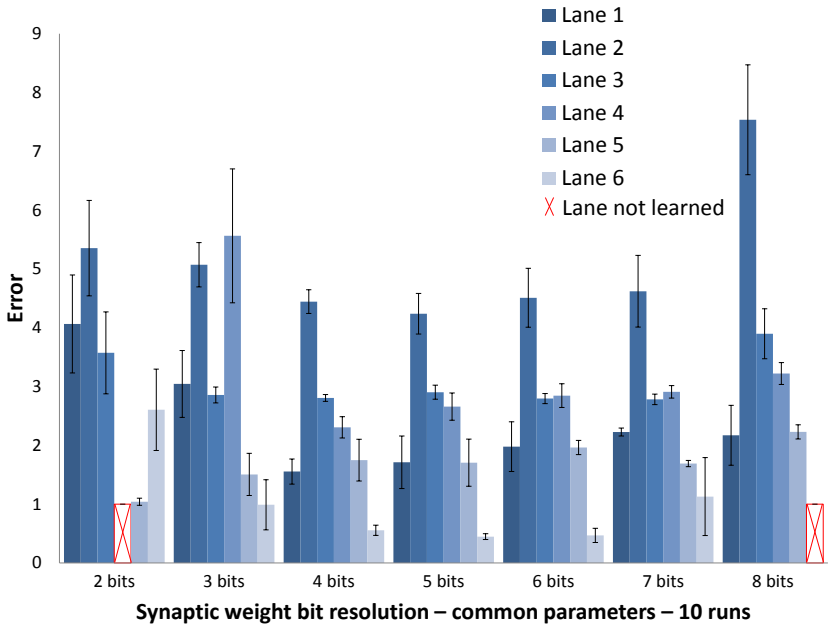


FIGURE 2.5 – Moyenne des erreurs pour chaque voie de circulation avec une résolution synaptique de 2 à 8 bits et des paramètres communs (table 2.3) sur 10 simulations.

Ces résultats montrent qu'il est possible de trouver un jeu de paramètres qui fonctionne pour chaque résolution synaptique (de 2 à 8 bits). Cela nous permettra, lors de nos implémentations futures de réseaux de neurones impulsionnels, de modifier la résolution des poids synaptiques sans apporter de modifications aux autres paramètres du réseau.

Dans la suite de nos travaux, nous utiliserons une résolution de poids synaptique de 4 bits. Cette résolution démontre un score d'erreurs équivalant aux résolutions supérieures et un score supérieur aux résolutions inférieures. Nous pouvons aussi noter que le réseau reste fonctionnel avec une résolution minimale de 2 bits.

2.5 Impact d'une décroissance linéaire du potentiel membranaire

Dans cette partie, nous étudions la possibilité de simplifier l'unité de traitement du réseau impulsionnel, le neurone. Ces réseaux utilisent un neurone de type «Leaky Integrate and Fire » et leur composante de fuite en est un élément majeur. La fuite du neurone permet d'intégrer un élément temporel dans l'intégration du neurone. En effet, sans fuite, le potentiel membranaire du neurone reflète l'ensemble de l'activité passé sans prendre en compte le temps écoulé. Avec une fuite, l'intégration du neurone ne reflète que l'activité dans un laps de temps passé, défini par la constante de fuite.

La fuite est généralement modélisée par une fonction décroissance exponentielle (voir section 1.2.2). Dans le contexte d'un simulateur évènementiel, l'équation régissant la fuite exponentielle d'un neurone au temps $t+\Delta t$ est donnée en équation 2.2, où t est le temps de la dernière mise à jour du calcul de fuite, τ_{Leak} est la constante de temps de fuite du neurone et w le poids synaptique à intégrer.

$$u_{t+\Delta t} = u_t . \exp\left(-\frac{\Delta t}{\tau_{leak}}\right) + w \tag{2.2}$$

L'implémentation d'un calcul de fuite exponentielle en analogique s'effectue en plaçant une résistance en parallèle avec le condensateur qui effectue l'intégration du neurone. Dans cette configuration, le courant de fuite dans la résistance suit une loi exponentielle. En numérique, l'implémentation d'un calcul exponentiel dans un ALU peut être très couteuse en surface et en énergie. Par exemple, l'implémentation d'une fonction exponentielle dans un ALU avec une précision virgule flottante nécessite un ajout de 3700 portes logiques (Vázquez and Antelo (2003)). Une astuce pour éviter ce surplus est d'utiliser une table de correspondance ("Look-Up Table", LUT), mais cela engendre une consommation d'éléments mémoires et d'accès mémoire. Dans Joubert et al. (2012), l'étude montre qu'une implémentation analogique permet d'avoir un gain en superficie ($\times\frac{1}{5}$) ainsi qu'en consommation d'énergie ($\times\frac{1}{20}$) pour une technologie CMOS 65 nm. Il est aussi montré que le gain en superficie serait nul pour une technologie sub-22 nm. Le remplacement du calcul exponentiel par un calcul linéaire peut diminuer la surface de l'ALU et le temps d'exécution du calcul de fuite et ainsi diminuer l'énergie totale consommée. L'équation régissant la fuite linéaire d'un neurone au temps $t + \Delta t$ est donnée en équation 2.3, où V_{leak} est la vitesse de la fuite en $seconde^{-1}$ (en considérant que l'intégration du neurone est une grandeur sans dimension).

$$u_{t+\Delta t} = max(0, u_t - \Delta t.V_{leak}) + w \tag{2.3}$$

Dans un environnement numérique, il y a deux manières pour effectuer la mise à jour de l'intégration du neurone, évènementielle ou synchrone.

Mise à jour synchrone Le potentiel membranaire (ou intégration) du neurone est mise à jour à un pas de temps régulier. Le terme Δt devient donc une constante ayant pour valeur T_{CLOCK}. Pour le calcul d'une fuite exponentielle, le terme exponentiel est supprimé et simplifié par une constante C_{exp}. Le calcul de fuite se résume donc à une simple multiplication (équation 2.4). Dans le cas d'une fuite linéaire, le calcul de fuite est équivalant à une simple soustraction (équation 2.5).

$$u_{t+\Delta t} = u_t.C_{exp} \tag{2.4}$$

$$u_{t+\Delta t} = max(0, u_t - C_{lin}) \tag{2.5}$$

Mise à jour évènementielle Dans ce cas, le calcul de fuite du neurone s'effectue lors de chaque potentiel d'action pré-synaptique. Δt est variable et doit être évalué par le réseau en implémentant un compteur de temps pour chaque neurone. Pour un calcul exponentiel, le calcul de la fuite est équivalant à deux multiplications et une exponentielle (équation 2.2). Pour un calcul linéaire, une multiplication et une soustraction doivent être effectuées (équation 2.3).

Nous pouvons souligner que peu importe le choix, pris par le concepteur du circuit, de la méthode de mise à jour du potentiel membranaire du neurone, un calcul de fuite linéaire sera toujours plus avantageux par rapport à un calcul exponentiel.

Nous avons simulé notre réseau avec des neurones intégrant un calcul de fuite linéaire (la méthode de mise à jour du potentiel membranaire dans XNET est évènementielle). Durant mes simulations, dans un but de diminution du temps d'exécution, nous avons remarqué qu'un réseau possédant 20 neurones de sortie au lieu de 60 démontrait des capacités d'apprentissage équivalentes. Le réseau simulé ici possède donc 20 neurones de sortie et une résolution des poids synaptiques de 4 bits. Les paramètres optimaux résultant de l'évolution génétique sont présentés dans le tableau 2.4. La moyenne des erreurs de chaque voie de circulation sur 100 simulations est présentée dans la figure 2.6. Ces résultats montrent un taux de reconnaissance moyen supérieur à 95%. *Nous pouvons donc affirmer que la capacité d'apprentissage d'un réseau de neurones impulsionnels, à base de neurone «Leaky Integrate and Fire » dont la composante de fuite est linéaire, n'est pas affectée, ce qui valide cette implémentation.*

TABLE 2.4 – Valeurs des paramètres neuronaux et synaptiques de la simulation avec une fuite linéaire figure 2.6.

$W_{\mathbf{len}}$ *(bits)*	I_{thres} $(/W_{\max})$	V_{leak} (ms^{-1})	T_{LTP} *(ms)*	$T_{inhibit}$ *(ms)*	T_{refrac} *(ms)*	$\Delta(w_+, w_-)$
4	573	112	7.5	53	531	(3,1)

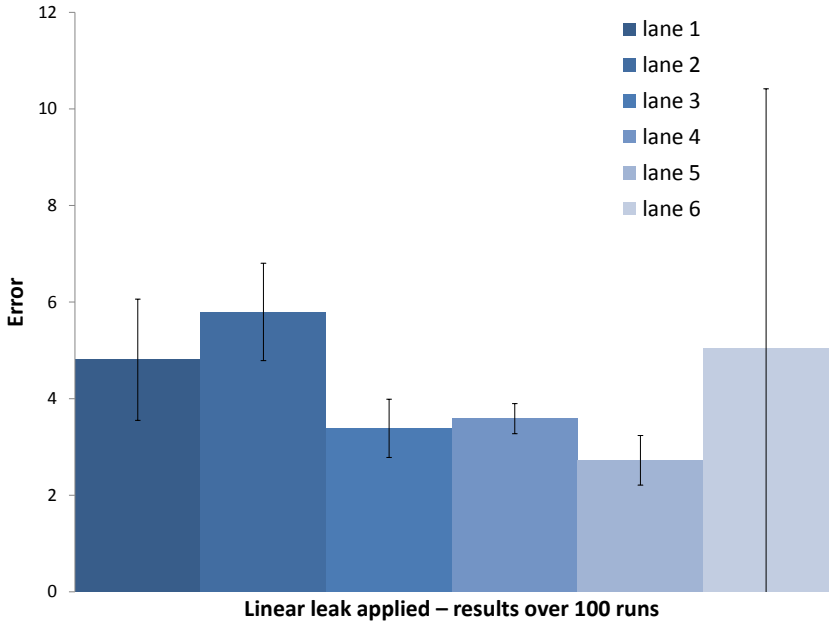


FIGURE 2.6 – Erreurs de chaque voie de circulation avec des neurones de sortie possédant une fuite linéaire.

2.6 Impact du remplacement des fenêtres temporelles par une FIFO

Spike Timing Dependant Plasticity (STDP) est un mécanisme d'apprentissage naturel récemment observé en biologie et considéré comme l'un des fondements de l'apprentissage dans le cerveau (Markram et al. (2011)). Nous pouvons rappeler que STDP est basé sur le temps d'arrivée des potentiels d'action pré-synaptique et post-synaptique. Pour cela, chaque neurone du réseau doit, d'une manière ou d'une autre, enregistrer le temps lorsqu'il émet un potentiel d'action. Pour un réseau numérique, cela implique d'implémenter un compteur de temps pour chaque neurone. Quand le seuil d'un neurone est atteint, il doit émettre un potentiel d'action (impulsion post-synaptique) puis doit comparer la valeur du compteur de temps de chaque neurone pré-synaptique. Les neurones, dont les compteurs ont une valeur inférieure à la constante de temps T_{LTP}, voient leurs synapses renforcées, et tous les autres neurones voient leurs synapses affaiblies.

Notre réseau impulsionnel de détection de véhicules possède 32 768 neurones d'entrée, cela signifie que nous devons implémenter 32 768 compteurs de temps, mais aussi, afin de pouvoir comparer ces temps à la constante T_{LTP}, il faut implémenter 32 768 comparateurs pour l'ensemble des neurones de sorties (car quand un neurone de sortie émet une impulsion, les autres sont en période d'inhibition) pour pouvoir effectuer les comparaisons en parallèle, ce qui augmente significativement la surface de silicium du réseau. Si les comparaisons sont effectuées en série, un seul comparateur est nécessaire, mais 32 768 coups d'horloges minimum seront exigés pour effectuer toutes les comparaisons nécessaires.

Dans ces travaux, notre hypothèse est que l'ordre d'arrivée des potentiels d'action pré-synaptiques est plus important que leur temps d'arrivée précis. Cette hypothèse a déjà été proposée (Thorpe (1990)) et est utilisée dans le simulateur SpikeNet (Thorpe et al. (2004)). Ainsi, nous proposons de remplacer la fenêtre temporelle de potentialisation des synapses T_{LTP} par une simple mémoire First-In First-Out (FIFO). Cette FIFO a une taille qui correspond au nombre d'évènements présents dans la fenêtre LTP. Les synapses présentes dans la fenêtre LTP ne dépendent donc plus du temps relatif entre les potentiels d'actions pré-synaptiques et post-synaptiques mais des synapses s'étant activées en dernière. Dans cette implémentation, à chaque activation neuronale pré-synaptique, l'adresse de ce neurone est ajoutée à la mémoire FIFO. Une seule FIFO peut être partagée par l'ensemble des neurones d'une même couche ce qui permet d'économiser de la surface CMOS si on compare à une implémentation basée sur le temps d'arrivée des potentiels d'actions. Quand le seuil d'un neurone est atteint, il émet un potentiel d'action puis renforce les synapses des neurones dont l'adresse est présente dans la FIFO et affaiblie toutes les autres synapses.

Avec cette implémentation, quand un neurone pré-synaptique émet deux potentiels d'actions dans une courte période de temps précédant un potentiel d'action post-synaptique, alors son adresse est présente deux fois dans la FIFO. Dans ce cas, deux possibilités s'offrent à nous. Soit la synapse est renforcée deux fois, soit la synapse n'est renforcée qu'une seule fois. Dans le second cas, il faudrait ajouter un module CMOS pour éviter les doublons dans la FIFO ce qui impliquerait une lecture de la FIFO à chaque nouveau potentiel d'action pré-synaptique avant d'insérer son adresse dans la FIFO. Cela engendrerait une consommation en surface de silicium et compromet donc l'optimisation recherchée dans notre étude. C'est pourquoi nous utilisons la première méthode. Nous ne cherchons pas à éliminer les doublons et renfonçons la synapse autant de qu'elles sont présentes dans la FIFO.

Pour intégrer une STDP basée sur l'ordre d'arrivée dans XNET, nous avons im-

plémenté une FIFO. Le paramètre de la fenêtre LTP temporelle T_{LTP} a été remplacé par le paramètre de la taille de la FIFO. Le réseau simulé est identique au précédent avec 20 neurones de sortie, des poids synaptiques de 4 bits et un terme de fuite exponentiel. Nous avons effectué une évolution génétique qui nous a permis de trouver les paramètres optimaux, présentés dans le tableau 2.5.

TABLE 2.5 – Valeur des paramètres optimaux résultant de l'évolution génétique du réseau avec une STDP basée sur l'ordre d'arrivée des potentiels d'actions.

$W_{\boldsymbol{len}}$ *(bits)*	I_{thres} $(/W_{\max})$	V_{leak} (ms^{-1})	***FIFO*** *(events)*	$T_{inhibit}$ *(ms)*	T_{refrac} *(ms)*	$\Delta(w_+, w_-)$
4	440	59	**1000**	56	578	(1,1)

Le résultat de l'évolution génétique montre que la taille optimale de la FIFO est de 1 000 évènements. Afin de mieux comprendre l'influence de la taille de la FIFO sur le score et pouvoir diminuer la taille de la FIFO jusqu'à un minimum où le score est encore satisfaisant, nous avons effectué plusieurs évolutions génétiques avec des tailles fixes de FIFO variant de 10 à 30 000 évènements. Les paramètres résultant de ces évolutions génétiques sont présentés dans le tableau 2.6. Le nombre moyen d'erreurs sur 100 simulations pour chaque voie de circulation avec des tailles de FIFO différentes est présenté dans la figure 2.7.

TABLE 2.6 – Valeur des paramètres optimaux, pour des tailles de FIFO variant de 10 à 30 000 évènements, résultant de l'évolution génétique du réseau avec une STDP basée sur l'ordre d'arrivée des potentiels d'actions des simulations de la figure 2.7.

$W_{\boldsymbol{len}}$ *(bits)*	I_{thres} $(/W_{\max})$	V_{leak} (ms^{-1})	***FIFO*** *(events)*	$T_{inhibit}$ *(ms)*	T_{refrac} *(ms)*	$\Delta(w_+, w_-)$
4	1261	0	**10**	8	97	(15,1)
4	154	228	**50**	107	666	(15,1)
4	15	114	**100**	40	547	(7,1)
4	202	54	**200**	118	544	(3,1)
4	440	59	**1000**	56	578	(1,1)
4	2370	71	**5000**	149	512	(1,5)
4	2713	109	**8200**	150	520	(1,5)
4	2246	112	**11 500**	240	582	(1,14)
4	3750	214	**30 000**	68	583	(1,15)

Ces résultats montrent que le réseau est fonctionnel avec des tailles de FIFO allant de 200 à 11 500 évènements. Sachant qu'une même synapse peut être présente plusieurs fois dans la FIFO, nous avons voulu savoir combien de synapses uniques, en moyenne, étaient renforcées lors de chaque cycle de programmation. Le tableau 2.7 présente le nombre moyen de synapses uniques étant renforcées et leur pourcentage par rapport au nombre total de synapses (32 768) pour chaque taille de FIFO. On peut remarquer, par exemple, qu'avec une taille de FIFO de 1000, seulement 673 synapses sont renforcées. Cela signifie que, sur ces 673 synapses, plusieurs ont été renforcées deux fois ou plus lors de chaque impulsion post-synaptique.

Afin de mieux visualiser l'influence de la taille de la FIFO sur l'apprentissage, une reconstruction des poids synaptiques après apprentissage d'un neurone spécialisé pour une voie de circulation a été effectuée pour des tailles de FIFO différentes. Les reconstructions sont présentées dans la figure 2.8. On peut voir qu'avec une taille de FIFO trop grande, le neurone commence à apprendre plusieurs voies de circulation ce qui nuit à l'apprentissage et explique le fort taux d'erreurs visible dans la figure 2.7.

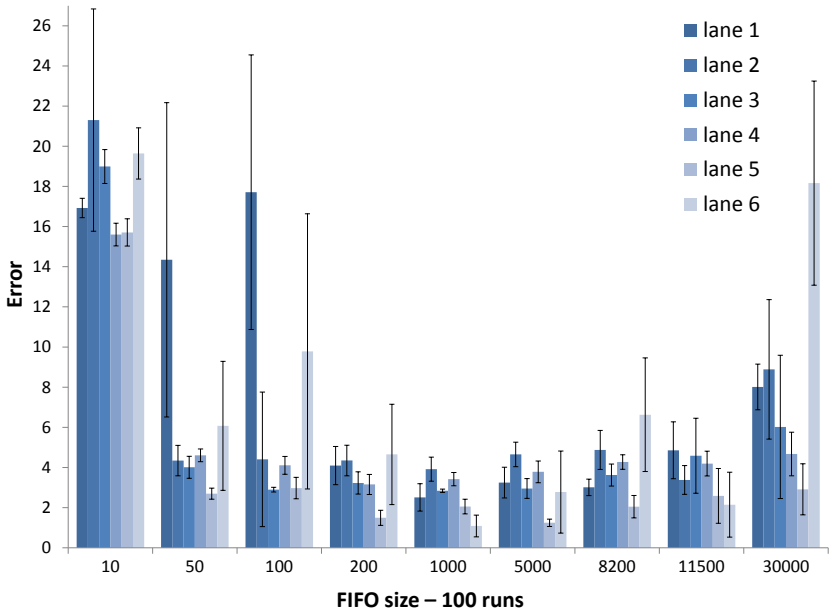


FIGURE 2.7 – Moyenne des erreurs par voie de circulation pour des tailles de FIFO variant de 10 à 30 000 évènements sur 100 simulations.

TABLE 2.7 – Nombre moyen et pourcentage de synapses uniques renforcées lors de chaque impulsion post-synaptique avec plusieurs tailles de FIFO.

taille de la FIFO *(nb évènements)*	*Synapses renforcées (nb synapses)*	*Synapses renforcées (%)*
10	9	0.03
50	49	0.15
100	97	0.30
200	180	0.55
1000	673	2.05
5000	2089	6.37
8200	2808	8.57
11 500	3432	10.5
30 000	6355	19.4

Pour intégrer ce type de réseau dans une implémentation numérique dédiée à l'embarqué, nous allons rechercher la taille de FIFO minimale tout en ayant une forte capacité d'apprentissage. Les résultats montrent qu'une taille de FIFO de 200 évènements est suffisante avec un taux de reconnaissance de véhicule de 97%. *Cela signifie qu'il est possible de remplacer les 32 768 compteurs de temps ainsi que les 32 768 comparateurs par une simple FIFO capable d'enregistrer 200 adresses AER de 15 bits partagées par l'ensemble des neurones de sortie.*

2.7 Résultat comprenant l'ensemble des modifications

Afin de valider un réseau intégrant l'ensemble des trois modifications décrites dans ce chapitre, nous avons effectué une dernière évolution génétique d'une simulation avec ces modifications pour une optimisation globale. Le réseau intègre donc des poids synaptiques de 4 bits, des neurones impulsionnels ayant un calcul de fuite linéaire et une STDP basée sur l'ordre d'arrivée des potentiels d'actions. L'architecture finale consiste donc en une seule couche de 20 neurones possédant chacun 32 768 synapses pour un

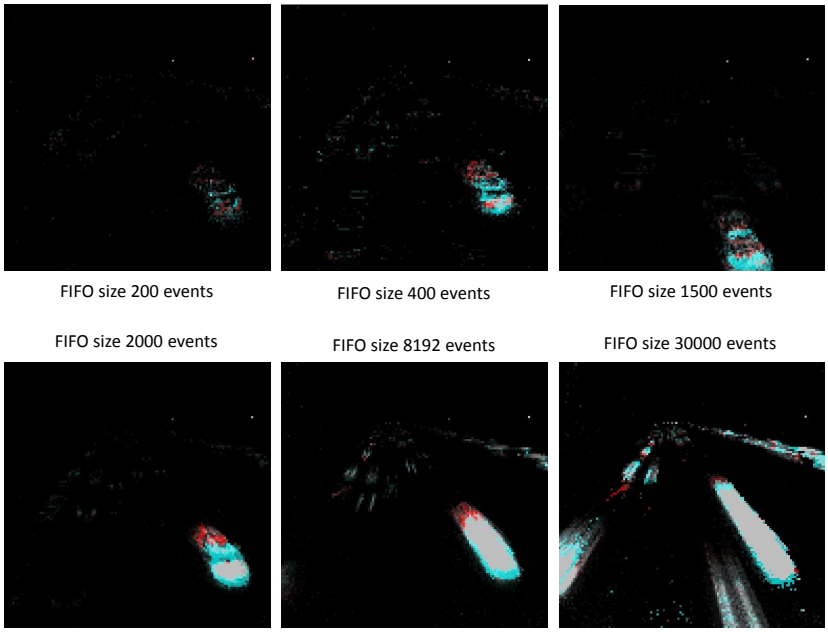


FIGURE 2.8 – Reconstruction des poids synaptiques après apprentissage d'un neurone pour des tailles de FIFO différentes. Une taille de FIFO excessive empêche un neurone de se spécialiser dans une seule voie de circulation.

nombre total de synapses de $128 \times 128 \times 2 \times 20$ ou 655 360 synapses. Les paramètres optimaux résultant de l'évolution génétique sont présentés dans le tableau 2.8. La figure 2.9 montre le nombre d'erreurs moyen sur 100 simulations pour chaque voie de circulation.

TABLE 2.8 – Valeurs des paramètres neuronaux et synaptiques optimaux de la simulation incluant les trois modifications.

$W_{\boldsymbol{len}}$ *(bits)*	I_{thres} $(/W_{\max})$	V_{leak} (ms^{-1})	***FIFO*** *(events)*	$T_{inhibit}$ *(ms)*	T_{refrac} *(ms)*	$\Delta(w_+, w_-)$
4	608	52	1012	67	530	(1,1)

Le résultat de cette simulation montre qu'avec l'implémentation de ces trois modifications, le taux de reconnaissance du réseau reste identique au réseau original de la figure 2.4, 98 %. *Ces modifications n'affectent pas l'apprentissage ou la fonctionnalité du réseau impulsionnel.*

2.8 Mises en place des optimisations

Durant l'année 2014, un retour d'expérience intéressant a mis en valeur les optimisations présentées dans ce chapitre. En effet, dans le cadre du stage de Vincent Lorrain dans notre laboratoire, nous lui avons demandé d'implémenter le réseau de neurones impulsionnels (présenté section 2.2) avec apprentissage STDP sur un FPGA. Comme nous l'avons vu, ce réseau, qui n'est pas d'une complexité extrême, possède tout de même 32 768 neurones d'entrée. Or, durant la mise en place du mécanisme d'apprentissage STDP, il s'est rendu compte qu'il n'est tout simplement pas possible d'instancier les 32 768 registres nécessaire à la mémorisation des dernières activations, car le FPGA ne possède pas assez d'éléments logiques. Il a donc mis en pratique les optimisations présentées dans ce mémoire, dans la section 2.6, à commencer par le remplacement

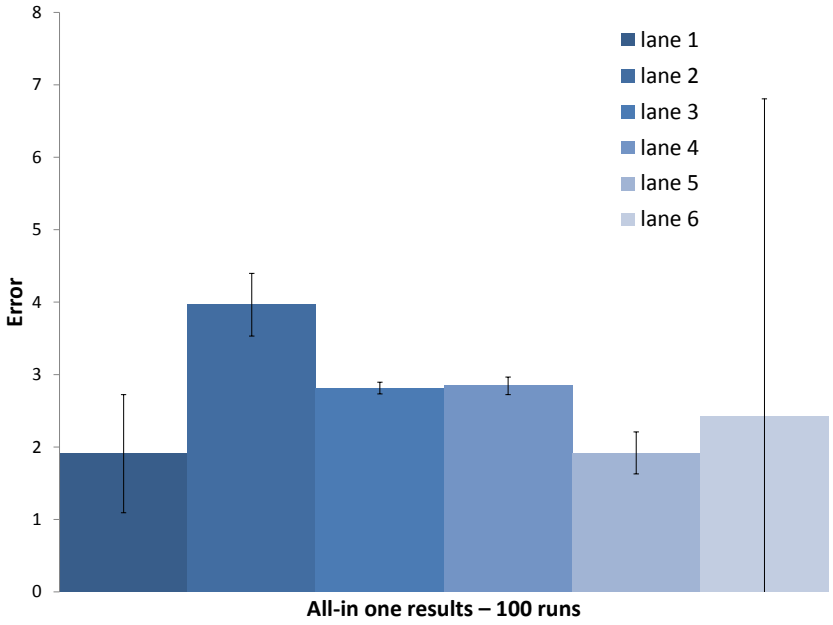


FIGURE 2.9 – Moyenne des erreurs pour chaque voie de circulation sur 100 simulations avec un réseau de 20 neurones de sorties, avec une résolution synaptique de 4 bits, des neurones incluant une fuite linéaire et une STDP par ordre.

de la notion temporelle de la règle STDP par la notion d'ordre. Il a donc implémenté une FIFO pour mémoriser les x dernières entrées activées, celles qui verront leur poids synaptique augmenté. De même, il a implémenté un neurone dont la composante de fuite suit une décroissance linéaire ainsi qu'une résolution synaptique de 4 bits. Le système créé a démontré sa capacité d'apprentissage sur une application analogue à celle présentée dans ce manuscrit, la détection de trajectoires. Ces travaux qui appuient les résultats présentés dans ce manuscrit feront l'objet d'une publication future.

2.9 Discussion et perspectives

Discussion Bien que nous ayons utilisé une résolution de 4 bits pour les poids synaptiques lors de nos différentes études, nous avons démontré que le réseau reste fonctionnel à partir d'une résolution synaptique de 2 bits (figure 2.4). Pour la reconnaissance dans le domaine de la vision, il existe deux types de problématiques distinctes qui sont la détection et la classification. Pour des problèmes de détection, comme dans l'exemple présenté ici, 2 bits suffiront pour détecter un véhicule circulant sur une autoroute. Néanmoins, si nous nous intéressons à reconnaitre un véhicule (forme/marque/type) alors un degré de détail plus élevé est nécessaire qui implique l'implémentation d'une résolution synaptique plus élevée. Des travaux (Querlioz et al. (2011)) ont montré que pour une application de classification de chiffres manuscrits (base de données MNIST), le taux de reconnaissance du réseau impulsionnel diminue considérablement quand le nombre de niveaux d'une synapse est inférieur à 20 niveaux (résolution synaptique de 4-5 bits).

Pour considérer cette résolution synaptique inférieure à 2 bits, une méthode de programmation stochastique doit être utilisée. En effet, sans programmation stochastique, le poids des synapses d'un neurone après apprentissage reflète uniquement le dernier cycle de programmation de celui-ci. Les synapses étant dans la fenêtre LTP sont à '1' et toutes les autres sont à '0'. Dans un environnement analogique et notamment avec des réseaux synaptiques utilisant des mémoires émergentes memristives, la stochasticité

peut être implémentée en prenant avantage de la stochasticité intrinsèque de ces composants avec des impulsions proche-programmation, d'un niveau de tension ou de durée inférieures à ce qui est requis pour assurer une commutation avec un taux de succès de 100%. Ces impulsions dites "faibles", ont une probabilité donnée de faire commuter (programmer) le composant permettant un apprentissage sur le long terme. Dans un environnement numérique, la stochasticité peut être implémentée par un générateur de nombre pseudo aléatoire pondéré. Quand une synapse est dans la fenêtre LTP ou LTD, le générateur détermine si la synapse effectue une plasticité ou non, de façon aléatoire, mais pondérée (probabilité de LTP et LTD).

Dans notre dernière étude, nous pouvons remarquer que la taille de la FIFO est directement proportionnelle avec la taille du motif à détecter. Pour notre application de détection de véhicule, les neurones doivent apprendre des motifs (véhicules) de même taille à une position donnée. C'est un cas presque parfait pour l'implémentation d'une STDP basée sur l'ordre d'arrivée des impulsions ayant une taille de mémoire FIFO prédéfinie. Si le neurone avait besoin d'appendre des motifs de taille différente alors le réseau ne serait pas aussi performant. L'utilisation de FIFO de taille différente pour une même couche de neurone, proportionnelle aux tailles des motifs à apprendre, est une solution envisageable restant à étudier.

Dans ce chapitre, nous avons fait un premier travail d'optimisation d'un réseau de neurones impulsionnels dédié à la vision, en vue de son intégration numérique dans un système embarqué. Cependant l'implémentation des synapses à base de mémoire classique de type FLASH, SRAM ou DRAM ne permet pas de recréer le parallélisme naturel des réseaux neuronaux, et est limité en termes de débit par le bus reliant la mémoire au système de calcul des neurones. Toutefois, la récente découverte et le rapide développement de technologies memristives pourraient implémenter efficacement une lecture et une programmation parallèle des synapses. Dans le chapitre suivant, nous présentons tout d'abord ces dispositifs avant de les comparer en fonction des qualités recherchées dans une synapse.

Chapitre 3

Synapses et Nanotechnologies : Mémoires émergentes

Sommaire

3.1 Introduction 43
3.2 Les dispositifs memristifs 44
3.2.1 Introduction 44
3.2.2 Les familles de dispositifs memristifs 45
3.3 Le dispositif memristif : la synapse idéale ? 46
3.3.1 Période de rétention 47
3.3.2 Résolution du poids synaptique 47
3.3.3 Cycle de programmations 49
3.3.4 Densité d'intégration 49
3.3.5 Méthode de programmation 50
3.4 Discussion et perspectives 51

3.1 Introduction

Dans les systèmes neuromorphiques, les synapses ont pour rôle de propager l'information d'un neurone à un autre en la pondérant du poids synaptique. Ainsi ma synapse doit mémoriser son poids synaptique qui devra pouvoir être modifié par le biais de mécanismes d'apprentissage. Dans les implémentations matérielles de systèmes neuromorphiques, deux grandes familles se distinguent : les implémentations numériques et les implémentations analogiques.

Avec une implémentation purement numérique, le poids synaptique est stocké de façon binaire dans une mémoire classique de type SRAM ou Flash pour être ensuite additionné à la valeur de l'intégration du potentiel membranaire grâce à une ALU (ex : Furber et al. (2013)).

En analogiques, différents éléments mémoire peuvent être utilisés pour stocker la valeur du poids synaptique : des condensateurs (Elias et al. (1997); Mahowald and Watts (1997)), des transistors à grilles flottantes (Frohman Bentchkowsky (1971)) ou encore de la mémoire classique RAM (Furber et al. (2013)).

Une des premières méthodes proposées est d'utiliser un condensateur pour stocker la valeur du poids synaptique analogique, mais cette solution nécessite des cycles de rafraichissement dûs à la fuite du condensateur sur une échelle de temps définie par la constante RC, où C est la capacité du condensateur et R la résistance de fuite. Ces

valeurs dépendent de la taille du condensateur, de la précision recherchée, de l'importance des courants de fuite, du bruit et de la température du système. Par exemple dans (Elias et al. (1997)), une fréquence de rafraichissement entre 12Hz et 200Hz est requise pour une précision entre six et dix bits.

Une seconde méthode consiste à utiliser des transistors à grilles flottantes (Diorio et al. (1996); Diorio et al.; Hindo (2014)). Cette technologie est capable de sauvegarder une valeur analogique très précisément pendant de longues périodes dues à son isolement électrique (pas de fuite). Par exemple, une précision sur huit bits peut être assurée pendant 8 ans (Srinivasan et al. (2008))

Enfin, une troisième méthode est d'utiliser une architecture DAC/ADC. Lors de la lecture, l'information binaire stockée dans une RAM classique est convertie par un DAC en signal analogique pour être ensuite intégrée dans le condensateur représentant le potentiel membranaire. Après un apprentissage, la nouvelle valeur synaptique est convertie par un ADC pour être mise à jour dans la RAM (Schemmel et al. (2010)).

Cependant, des dispositifs mémoire émergents pourraient permettre l'implémentation de synapses de façon efficace, grâce à leurs fortes densités intégrations, leur non-volatilité et leur faible consommation d'énergie. Dans ce chapitre nous allons tout d'abord présenter ces différentes technologies émergentes, puis nous allons les comparer en fonction des qualités recherchées dans une synapse qui sont 1) la période de rétention de l'information, 2) le caractère multi-niveaux, 3), l'endurance, 4) la densité d'intégration et enfin 5) la méthode de programmation.

3.2 Les dispositifs memristifs

3.2.1 Introduction

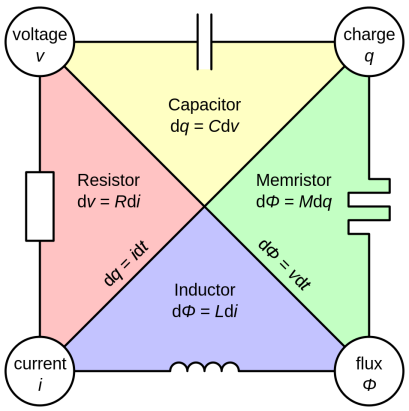


FIGURE 3.1 – Relation non-linéaire courant (i) - tension (v) de la résistance, charge électrique (q) - tension (v) du condensateur, courant(i) - flux électrique (ϕ) de l'inductance et charge électrique (q)- flux électrique (ϕ) du memristor.

Tout comme la résistance, le condensateur ou l'inductance, le dispositif memristif fait partie de la catégorie des composants passifs non-linéaire fondamentaux de l'électronique (Chua (1971)). Matérialisé en 2008 dans les laboratoires Hewlett-Packard (Strukov et al. (2008)), il fut pourtant décrit dès 1971 par Léon Chua (Chua (1971)). Remarquant la relation non-linéaire courant (i) - tension (v) de la résistance, charge électrique (q) - tension (v) du condensateur et courant(i) - flux électrique (ϕ) de l'inductance (figure 3.1), il extrapola un quatrième composant mettant en relation le flux électrique (ϕ) et la charge électrique (q) qu'il appela Memristor (abréviation pour « memory resistor »). Les memristors, ou plus généralement les dispositifs memristifs, sont

des composants passifs dont la résistance varie en fonction de l'historique des courants circulant dans les dispositifs. La valeur de la tension aux bornes d'un dispositif memristif est donc dépendante du courant traversant le dispositif et de la valeur instantanée de sa memristance. Elle est alors modélisée par l'équation :

$$V(t) = M(q(t)).i(t) \tag{3.1}$$

Avec $i(t) = \frac{dq}{dt}$ le courant circulant dans le memristor, $v(t)$ la tension à ses bornes et $M(q(t)) = \frac{d\phi(t)}{dq(t)}$ sa valeur de memristance à l'instant t.

En fonction des mécanismes utilisés pour stocker l'information, il est possible de classer les dispositifs memristifs en plusieurs catégories. Dans ce chapitre, nous allons en distinguer trois parmi celles qui sont les plus matures (ITRS (2013)) : les STT-MRAM (Spin Transfer Torque-Magnetoresistive RAM), les PCRAM (Phase-Change RAM) et les RRAM (Resistive RAM).

3.2.2 Les familles de dispositifs memristifs

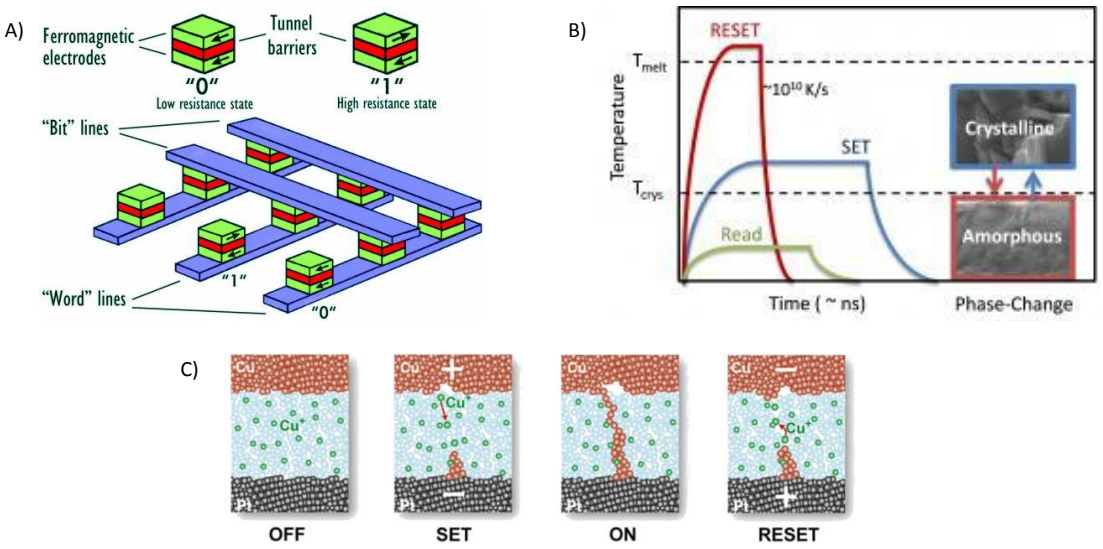


FIGURE 3.2

Spin Transfer Torque-Magnetoresistive RAM La technologie STT-MRAM est une mémoire magnétique non-volatile (Tehrani et al. (1999); Gallagher and Parkin (2006); Engel et al. (2005)). Contrairement aux mémoires classiques, l'information (bit) n'est pas stockée sous forme de charge électrique, mais d'orientation magnétique. La cellule MRAM est formée de deux couches ferromagnétiques séparées par un isolant. L'une des couches ferromagnétiques est constamment polarisée alors que l'orientation magnétique de la seconde peut être modifiée. La commutation de ces dispositifs s'effectue en changeant le spin des électrons, notamment par un effet tunnel (figure 3.2 A). La valeur de la cellule est déterminée en comparant la résistance électrique de celle-ci à une résistance de référence. Si les deux couches ferromagnétiques ont la même polarité alors la valeur résistance est plus faible que lorsque les polarités sont opposées.

Phase Change RAM Les dispositifs PCRAM (ou PCM) sont des mémoires à changement de phase non-volatiles (Pirovano et al. (2004); Wong et al. (2010b); Lai (2003)). Ils utilisent la propriété du chalcogénure qui, sous l'effet de la chaleur par effet Joule, peut prendre une forme amorphe (faiblement conducteur) ou cristalline (fortement

conducteur) (figure 3.2 B). La forme amorphe représente l'état de forte résistance (valeur binaire '0') et la forme cristalline représente l'état de faible résistance (valeur binaire '1').

Resistive RAM Les dispositifs RRAM sont basés sur la création d'un ou plusieurs ponts conducteurs dans un matériau isolant entre deux électrodes de métal (structure Metal-Insulator-Metal (MIM)) (figure 3.2 C). En fonction du mécanisme résultant de la création du filament conducteur, on peut discerner deux classes de dispositifs RRAM.

Conductive Bridge RAM Les dispositifs CBRAM, aussi appelés programmable metallization cell (PMC), sont composés d'une couche électrolyte solide entourée de deux électrodes, une inerte (ex : Tungstène) et une active électrochimiquement (principalement Argent, Cuivre). La migration de cations provenant de l'électrode active (ex : Ag^{+},Cu^{+}, ou Ca^{2+}) est à l'origine de la formation et de la destruction du filament conducteur dans l'électrolyte. Plusieurs matériaux peuvent être utilisés dans la couche électrolyte, notamment du $GeS2$ (Kund et al. (2005); Jameson et al. (2013); Bruchhaus et al. (2009)), du Ge_xSe_y (Yu and Wong (2011); Rahaman et al. (2012)), du TiO_2 (Abe et al. (2008); Tsunoda et al. (2007)), et du SiO_2 (Yang et al. (2012); Schindler et al. (2007); Balakrishnan et al. (2006)).

Oxide Resistive RAM Les dispositifs OxRAM sont basés sur le déplacement d'ions oxygène provenant de la couche d'oxyde pour créer le filament conducteur entre deux électrodes inertes (Waser and Aono (2007)). Ce déplacement d'ions vers une électrode, créé une lacune en oxygène dans l'isolant le rendant conducteur (valeur logique '1') et le déplacement inverse détruit le filament (valeur logique '0'). Plusieurs matériaux ont prouvé leur efficacité, par exemple le HfO_2 (Lee et al. (2008)), le NiO (Seo et al. (2004)), ou le Cu_2O (Fang et al. (2006)).

3.3 Le dispositif memristif : la synapse idéale ?

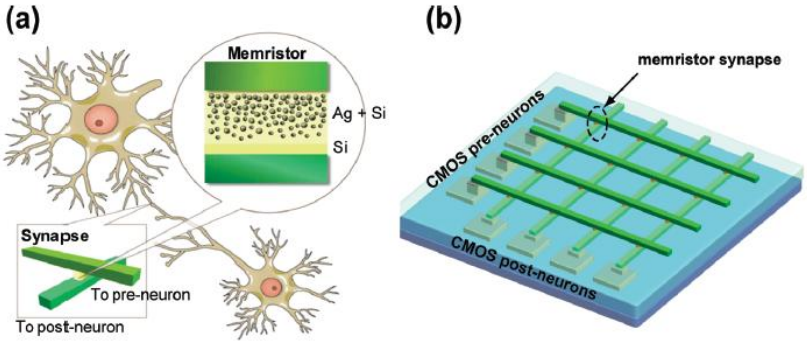


FIGURE 3.3 – a) Exemple de dispositif memristif implémentant une synapse. b) Exemple d'un crossbar memristif implémentant un champ synaptique entre des neurones pré-synaptiques et des neurones post-synaptiques (Jo et al. (2010))

La première question que nous pouvons nous poser est la suivante : y a-t-il une synapse idéale ? Une première constatation des neurosciences nous apprend qu'il existe deux familles de synapses, les chimiques et les électriques (Purves (2008)). La synapse électrique connecte un neurone à un autre neurone et peut transmettre un signal électrique de l'un à l'autre de façon bidirectionnelle et très rapide (instantanément comparé au délai imposé par la synapse chimique) (Purves (2008)). La synapse chimique connecte aussi un neurone à un autre neurone, mais utilise des neurotransmetteurs pour transmettre l'information. Plus d'une centaine de neurotransmetteurs ont été identifiés,

certain ont un effet excitateur sur le neurone post-synaptique et d'autres inhibiteurs, certain augmente la vitesse de la réponse électrique et d'autre la diminue (Purves (2008)). Les neurosciences nous apprennent donc qu'il n'existe pas de synapse idéale, mais une multitude.

Dans cette partie, nous allons décrire les caractéristiques principales que nous souhaitons retrouver dans une synapse artificielle qui permettent notamment la transmission de l'information, la pondération de l'information ainsi que la mise en place du mécanisme d'apprentissage STDP.

3.3.1 Période de rétention

La synapse est tout d'abord un élément mémoire, un dispositif capable de retenir une information sur le long terme. Il est aussi intéressant, pour une économie d'énergie que ce dispositif n'exige pas de rafraichissement comme pour la DRAM ou le maintien de l'alimentation comme pour la SRAM.

La MRAM, PCRAM et RRAM sont des mémoires non-volatiles qui répondent à ces exigences. La période de rétentions pour chacun de ces dispositifs est supérieure à dix ans (ITRS (2013)).

3.3.2 Résolution du poids synaptique

Une seconde caractéristique recherchée est le caractère multi-niveau du poids synaptique. Nous avons en effet besoin de plusieurs niveaux analogiques, mais la résolution précise est une question qui dépend de plusieurs facteurs. En effet, on peut distinguer quatre environnements afin de spécifier les besoins, en nombre de niveaux analogiques ou en nombre de bits numériques, du poids synaptique : La méthode d'apprentissage, soit en ligne, soit hors ligne et l'application visée, la détection ou la reconnaissance (table 3.1).

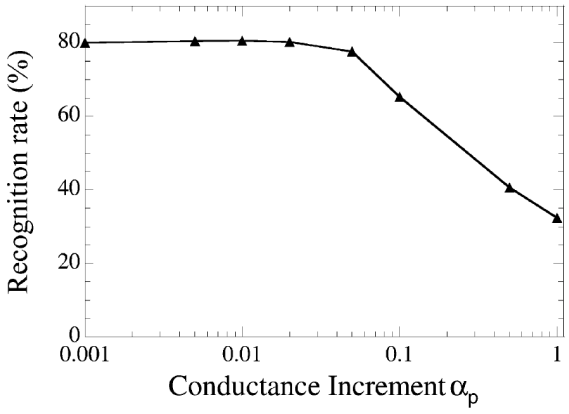


FIGURE 3.4 – Impact de la résolution synaptique sur le taux de reconnaissance (benchmark MNIST). Le taux de reconnaissance est maximum avec des résolutions synaptiques supérieures à 20 niveaux. (Querlioz et al. (2013))

En fonction de l'application, le nombre de niveaux synaptique dépend de la complexité de la tâche à accomplir. Pour une application de détection, déterminer par exemple si une voiture est présente dans l'image, alors le nombre de niveaux requis peut être faible (aux minimum deux niveaux analogiques ou un bit numérique) (Roclin et al. (2013)). Pour une application de reconnaissance, il faut être capable par exemple de déterminer le modèle de la voiture détecté. Dans ce cas, une résolution plus importante est requise ce qui permettra de mettre en valeur certains détails de l'image. Le

nombre de niveaux dépend aussi de la manière dont l'apprentissage STDP s'effectue, en ligne ou hors ligne .

Durant un apprentissage en ligne, plusieurs points sont à retenir. Premièrement, il n'est pas possible d'avoir seulement deux niveaux sans implémenter un apprentissage stochastique (voir chapitre 4 section 4.2.2), sans lequel, le poids synaptique ne reflète que le dernier mécanisme de LTP ou de LTD de la STDP. Trois niveaux au minimum sont donc requis pour effectuer un apprentissage non-stochastique. Ce nombre augmente en fonction de la complexité du motif à apprendre. Par exemple, dans ces travaux (Querlioz et al. (2013)), les auteurs montrent que le taux de reconnaissance résultant de l'apprentissage est maximum avec une résolution synaptique supérieure à vingt niveaux (ou cinq bits) sur la base de données MNIST de reconnaissance de chiffres manuscrits. On peut aussi voir sur cet exemple que l'augmentation de la résolution ne permet pas d'atteindre un taux de reconnaissance supérieur.

Après apprentissage, la résolution du poids synaptique dépend de l'application et du degré de complexité du motif à reconnaitre.

TABLE 3.1 – Ordre de grandeur de la résolution synaptique en fonction du mode d'apprentissage et de l'application.

		Apprentissage	
		En ligne	Hors ligne
Application	Détection	>10	>2
	Reconnaissance	>100	>20

On peut différencier deux classes de dispositifs memristifs, d'un côté les dispositifs binaires (deux niveaux analogiques ou un bit) et d'un autre, les dispositifs capables de stocker plusieurs niveaux analogiques (Multi-Level Cell (MLC)) ou plusieurs bits numériques. Un état de l'art de la capacité MLC des memristors est présenté ici Bichler (2012).

La CBRAM ainsi que l'OxRAM sont des dispositifs MLC, mais seulement en contrôlant le courant circulant dans le dispositif lors d'un SET ou d'un RESET (compliance current). En effet pour obtenir un SET gradué (ou un RESET gradué) il faut augmenter graduellement le courant durant un SET (ou un RESET) afin de contrôler de manière précise la formation (ou la destruction) du filament conducteur. Le contrôle du courant est réalisé à l'aide d'un transistor en série avec chaque CBRAM. Le courant est proportionnel à la tension de grille du transistor. Pour bénéficier du MLC dans le cas d'un apprentissage en ligne, il faut réaliser chacune des étapes suivantes séquentiellement lors de chaque cycle de programmation :

1. Lire la résistance d'une CBRAM
2. Augmenter (en cas de LTP) ou diminuer (en cas de LTD) la tension de grille du transistor en fonction de la valeur précédente de la CBRAM
3. Envoyer une impulsion de programmation

Ces étapes remettent directement en cause la programmation parallèle recherchée avec les dispositifs memristifs. Les dispositifs CBRAM et OxRAM sont donc utilisés de façon binaire. Pour émuler une synapse avec apprentissage en ligne possédant plus de deux niveaux logiques, il faut implémenter autant de CBRAM que de niveaux logiques souhaités par synapse ainsi qu'un apprentissage stochastique (apprentissage qui ne nécessite pas de cycle de lecture avant la programmation).

Le dispositif MRAM est un dispositif binaire et peut donc s'utiliser de la même manière que les dispositifs CBRAM et OxRAM binaire.

Le dispositif PCM peut réaliser une cristallisation (SET) graduée, mais le passage en phase amorphe (RESET) est un phénomène abrupt. La PCM possède aussi un problème de dérive de la résistance, quand elle est utilisée en MLC la cellule tend vers l'état de haute résistance (amorphe) (Suri et al. (2013a)). Ce dispositif peut donc être utilisé de deux manières différentes, soit de façon binaire, comme la CBRAM, soit en utilisant un système de deux PCM par synapse (Bichler et al. (2012b); Suri et al. (2011)).

3.3.3 Cycle de programmations

Lors d'un apprentissage hors ligne, la synapse n'est programmée qu'une seule fois avant l'utilisation du réseau de neurones dans son environnement. Dans le cas d'un apprentissage en ligne, chaque synapse doit pouvoir être programmée (SET, RESET) plusieurs fois, notamment dans le cas d'architecture de réseaux convolutionnels où les synapses sont partagées. Dans Garbin et al. (2014), les auteurs montrent une moyenne de $4,2 \times 10^5$ cycles de SET et $2,2 \times 10^5$ cycles de RESET par synapse pour une application de reconnaisse de chiffre manuscrit (base MNIST) avec une architecture de type réseau convolutionnels.

Comparer à la mémoire traditionnelle de type FLASH dont l'endurance est inférieure à 10^5 cycles (ITRS (2013)), les dispositifs memristifs cités dans ce chapitre ont une forte endurance et permettent réaliser un grand nombre de cycles d'écriture, par exemple 10^{10} pour la CBRAM (Kozicki et al. (2005)), 10^{12} pour l'OxRAM (Kim et al. (2011)), 10^8 pour la PCM (Hongxin et al. (2012)) et supérieur à 10^{15} pour la MRAM (Huai et al. (2011)).

3.3.4 Densité d'intégration

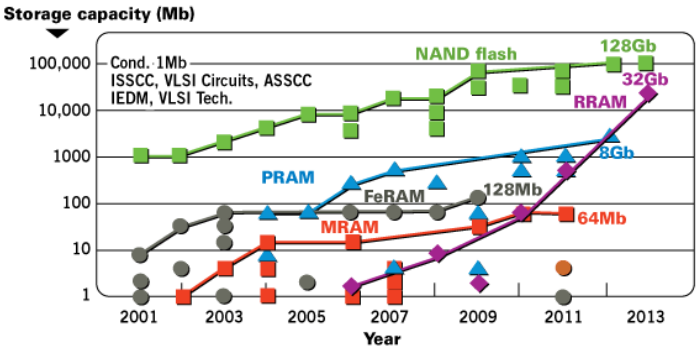


FIGURE 3.5 – Évolution de la capacité des mémoires non-volatiles (ISS (2013))

Le nombre de synapses dans un système neuromorphique impulsionnel peut être élevé. Prenons l'exemple d'un réseau complètement connecté dont les entrées correspondent à la caméra AER DVS128 de 128×128 pixels (Bichler et al. (2012a)). Dans cet exemple, chaque neurone est alors connecté aux 128^2=16 384 pixels de la caméra. Ce type de réseau nécessite donc 16 384 synapses par neurones pour une caméra dont la résolution n'est pas très importante. On peut estimer un nombre de synapses compris entre 1 et plusieurs centaines de millions Wan et al. (2013); Schmidhuber (2012); Ciresan et al. (2011). Il est donc important que la taille de la cellule synaptique soit la plus minimale possible pour un gain en surface. Nous avons répertorié (tableau 3.2) un état de l'art des différentes technologies mémoires émergentes afin de les comparer aux mémoires traditionnelles (DRAM-FLASH-SRAM) (données extraient de ITRS (2013)). Dans ce tableau, nous pouvons voir que la taille des cellules des mémoires émergentes est du même ordre de grandeur que la mémoire FLASH. La mémoire FLASH possède

des problèmes de conception pour les technologies inférieures à 22 nm (notamment la variation du seuil du transistor (ITRS (2013))) mais sa densité d'intégration continue d'augmenter principalement grâce au développement des cellules MLC ainsi qu'aux structures 3D (3D stacking). Contrairement aux autres dispositifs, les valeurs d'intégration de la CBRAM et de l'OxRAM sont expérimentales et ne prends pas en compte l'intégration d'un transistor de sélection. La figure 3.5 montre que les mémoires émergentes ont une évolution plus rapide que la mémoire FLASH et qu'elle est proche d'être rattrapée par la mémoire RRAM.

TABLE 3.2 – Taille des cellules des technologies mémoires non-volatiles traditionnelle (FLASH) et émergente (CBRAM-OxRAM-PCM-STT-MRAM). (ITRS (2013))

	Half pitch f (nm)	Facteur a	Aire de la cellule $= a \times f^2$
NAND FLASH (SLC)	18	4	1200
NAND FLASH (MLC)	18	1.3	420
STT-MRAM	65	20	84500
CBRAM	20	4	1600
OxRAM	5	4	100
PCM	45	4	8100

Ces informations montrent que l'utilisation future de mémoires émergentes en tant que synapses permettra de diminuer le coût silicium d'un réseau de neurones tout en permettant la mise en place de cycles de lecture et programmation hautement parallèle.

3.3.5 Méthode de programmation

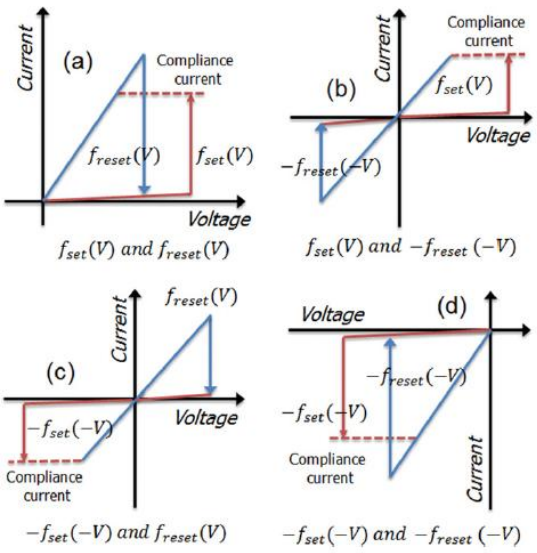


FIGURE 3.6 – Relation Tension-Courant de dispositifs bipolaires (b) et (c) et unipolaires (a) et (d) (Jeong et al. (2012))

Il est possible de classer les mémoires émergentes en deux catégories, les technologies unipolaires et les technologies bipolaires. Les dispositifs bipolaires effectuent un SET quand une tension positive est appliquée à ses bornes et effectue un RESET quand une tension négative est appliquée (ou vice versa) (figure 3.6 (b) et (c)). Les dispositifs

unipolaires ne sont programmés qu'avec des tensions positives, ils possèdent deux seuils positifs (ou deux négatifs), un pour le SET, et un pour le RESET (figure 3.6 (a) et (d)).

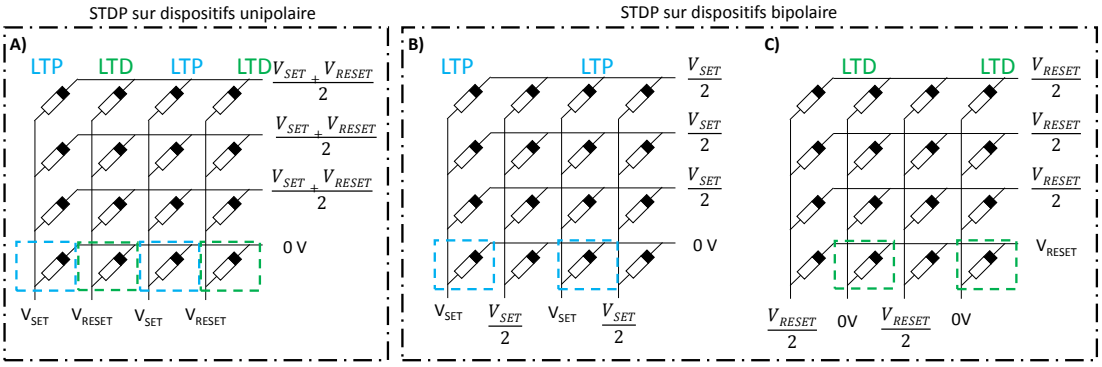


FIGURE 3.7 – Méthode de programmation dans un crossbar de dispositifs unipolaires et bipolaires. A) On remarque que le cycle de programmation (SET (LTP) et RESET (LTD)) des dispositifs unipolaire peut s'effectuer en un seul cycle alors que les dispositifs bipolaires ont besoin de deux cycles consécutifs B) SET (LTP) et C) RESET (LTD). Les transistors de sélection sont optionnels pour les dispositifs bipolaires si $|V_{th+} - V_{th-}| < min(|V_{th+}|, |V_{th-}|)$ (Bichler et al. (2013a)).

Afin de comparer ces deux mécanismes de commutation, étudions leur comportement lors d'une phase d'apprentissage STDP dans un crossbar. La figure 3.7 A) montre un exemple de cellules unipolaires réalisant un cycle de programmation STDP, on remarque qu'il est possible de réaliser la LTP (V_{SET}) et la LTD (V_{RESET}) pendant un seul et même cycle (par exemple V_{SET}=1,5 V et V_{RESET} = 3,0V). Les figures 3.7 B) et C) montrent respectivement les cycles de LTP (+1,5V) et LTD (-1,5V). On remarque que deux cycles successifs sont nécessaires pour les dispositifs bipolaires (Bichler et al. (2013a)).

On peut conclure que les dispositifs unipolaires peuvent présenter un avantage en termes de rapidité d'exécution lors d'un cycle d'apprentissage STDP. Il faut tout de même retenir que les deux types de dispositifs, bipolaires et unipolaires, permettent la programmation parallèle des synapses comparées aux mémoires classiques (FLASH-SRAM-DRAM)

3.4 Discussion et perspectives

TABLE 3.3 – Tableau comparatif des technologies mémoires non-volatiles

	FLASH	STT-MRAM	PCM	CBRAM	OxRAM
Rétention (années)	>10	>10	>10	>10	>10
Résolution	MLC	SLC	SLC	SLC	SLC
Endurance	10^5	10^{15}	10^8	10^{10}	10^{12}
Densité (nm^2)	420	84500	8100	1600	100
uni-bipolaire	uni	bi	uni	bi	bi

Dans ce chapitre, nous avons dans un premier temps introduit les dispositifs mémoires émergents et notamment trois familles de ces dispositifs, la mémoire magnétique (STT-MRAM), la mémoire à changement de phase (PCM) et la mémoire résistive

(RRAM) qui est composée de la mémoire à pont conducteur (CBRAM) et de la mémoire à oxyde (OxRAM). Dans un second temps, nous avons déterminé les caractéristiques recherchées dans une synapse artificielle.

Tout d'abord, nous avons montré qu'il est intéressant pour une question d'énergie que cette synapse puisse conserver l'information sans alimentation électrique et pendant un temps relativement long. Les dispositifs présentés dans chapitre répondent favorablement à cette caractéristique puisque ce sont des dispositifs non-volatiles et que leur période de rétention est supérieur à 10 ans, même ordre de grandeur que la mémoire traditionnelle FLASH.

Ensuite, nous avons montré que pour réaliser l'apprentissage d'un pattern complexe (visage, chiffre, etc.), cette synapse a besoin d'avoir plusieurs niveaux logiques ; or certains dispositifs possèdent intrinsèquement cette caractéristique par exemple la PCM.

Lors d'un apprentissage, une synapse doit être capable de modifier son poids de nombreuses fois ($> 10^5$). Nous avons montré que les dispositifs memristifs possèdent ce type d'endurance, exprimé en cycles de programmation. Les dispositifs présentés ici montrent une endurance de l'ordre de $10^{8\approx11}$ ce qui est intéressant comparée à la mémoire FLASH qui possède une endurance de l'ordre de 10^5 cycles de lecture/écriture.

Enfin, nous avons aussi vu que les mémoires émergentes pouvaient être intégrées de manière dense, comparable a la mémoire FLASH pour certain (CBRAM) et meilleur pour d'autre (PCM), tout en permettant une programmation parallèle de l'ensemble des dispositifs d'un même neurone ce qui n'est pas envisageable avec de la mémoire FLASH classique.

Dans ce chapitre, nous avons montré que les dispositifs mémoires émergents possèdent les qualités requises pour implémenter une synapse répondant aux besoins décrits précédemment. Hormis ces dispositifs mémoires, seules les structures à base de transistors à grille flottante Hasler and Marr (2013) pourraient implémenter efficacement des synapses. En effet, cette technologie est mature, dense et est capable de stocker 4 bits (ou 16 niveaux logiques) dans un seul dispositif (Li et al. (2008); Marotta et al. (2010)). Cependant, ces dispositifs posent des problèmes au niveau du passage à l'échelle pour les nœuds technologiques inférieurs à 22 nm (ITRS (2013)).

Les dispositifs memristifs, dont le développement est très actif dans beaucoup de laboratoires, ont de leurs côtés montrés une meilleure mise à l'échelle, par exemple l'OxRAM (100 nm^2) intégré avec des nanotubes de carbone. Ces dispositifs pourront sur le long terme montrer des caractéristiques bien plus intéressantes que les transistors à grille flottante. C'est pourquoi dans le chapitre suivant, nous allons étudier différentes structures de dispositifs memristifs CBRAM afin d'étudier les contraintes et avantages de ces dispositifs au niveau circuit.

Chapitre 4

Mémoire synaptique à base de dispositifs memristifs

Sommaire

4.1 Introduction 53
4.2 Modélisation VHDL-AMS d'une cellule CBRAM 55
4.2.1 Modèle CBRAM 55
4.2.2 Mémoire synaptique à base de dispositifs memristifs CBRAM 57
4.3 Structure de CBRAM en crosspoint 58
4.3.1 Crossbar 59
4.3.2 Matrice connectée par les anodes 65
4.3.3 Matrice connectée par les cathodes 69
4.3.4 Matrice – Surcoût associé au transistor de sélection dans une matrice 70
4.3.5 Conclusions des études réalisées 70
4.4 Modélisation d'un réseau de neurones impulsionnels et simulation VHDL-AMS 71
4.4.1 Réseau de neurones impulsionnels en VHDL-AMS 71
4.4.2 Simulation du réseau 75
4.5 Discussion et perspectives 76

4.1 Introduction

L'utilisation de mémoires classiques de type FLASH ou SRAM pour implémenter les synapses ne permet pas la lecture ou la programmation parallèle des synapses et est limitée en termes de débit par la taille du bus reliant la mémoire à l'unité de calcul. Lors de la lecture (activation d'un neurone d'entrée), le réseau doit lire dans la mémoire la valeur de l'ensemble des synapses du neurone activé afin d'additionner ces valeurs synaptiques aux valeurs d'intégration des neurones de sortie. Lors de la programmation (activation d'un neurone de sortie), le réseau récupère dans la mémoire les valeurs des synapses présentes dans la fenêtre LTP pour les incrémenter de la valeur $W+$ puis mettre à jour les nouvelles valeurs synaptiques dans la mémoire et enfin faire de même pour les synapses présentes dans la fenêtre LTD (décrément de $W-$). Cela engendre, dans les deux cas, un grand nombre d'accès de lecture et d'écriture dans la mémoire qui est un facteur important de limitation de taille de réseau. Or, les technologies mémoires de type memristives pourraient permettre l'intégration de ce parallélisme directement au cœur de la mémoire. La figure 4.1 illustre une manière d'implémenter un réseau

impulsionnel à base de dispositifs memristifs. Dans cet exemple, le réseau possède trois neurones d'entrée et deux neurones de sortie reliés entre eux par six synapses.

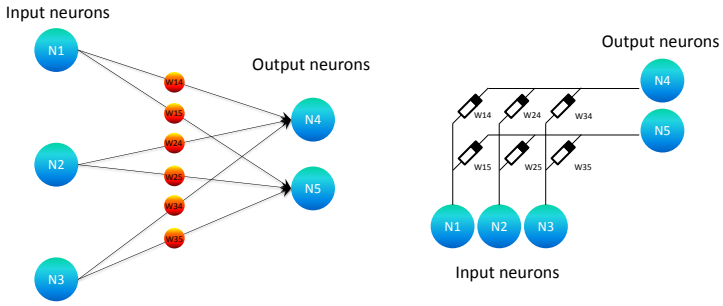


FIGURE 4.1 – Exemple d'implémentation d'un réseau de neurones impulsionnels à base de dispositifs memristifs.

La figure 4.2 illustre les cycles de lecture et de programmation du réseau impulsionnel. La lecture (figure 4.2 a)) interviens lors de l'activation d'un neurone d'entrée. Dans cet exemple, le neurone N1 s'active et émet une impulsion de lecture pondérée en parallèle par ses synapses, W14 et W15. Avec ce type d'architecture, la pondération d'une synapse est matérialisée par la conductivité d'un dispositif. Les courants résultants sont alors intégrés en parallèle par les neurones de sorties. La programmation (figure 4.2 b)) interviens quand un neurone de sortie atteint son seuil d'activation, N4 dans cet exemple. Il émet alors une impulsion de programmation, et programme en parallèle ses propres synapses, W14 W24 W34. Les mémoires émergentes peuvent donc nous permettre de mettre en place des cycles de lecture et de programmation parallèle et ainsi implémenter de larges réseaux impulsionnels sans être limitées par les accès mémoire tout en ne consommant qu'une très faible énergie. Le parallélisme naturel des réseaux neuronaux peut donc être retrouvé grâce à ces technologies.

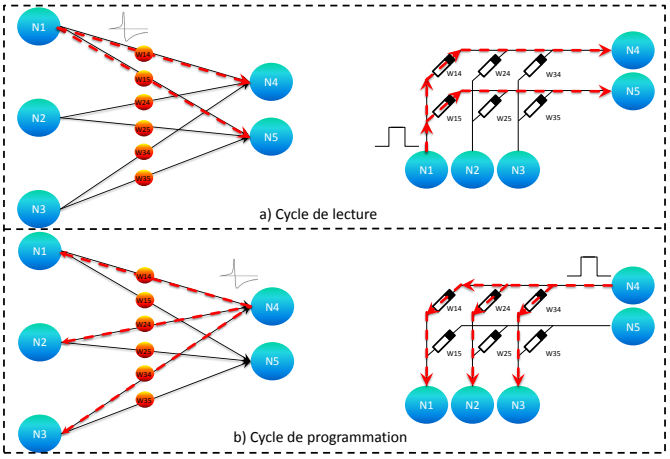


FIGURE 4.2 – a) Cycle de lecture, les dispositifs memristifs permettent une lecture parallèle des synapses d'un neurone d'entrée. B) Cycle de programmation, les dispositifs memristifs permettent une programmation parallèle des synapses d'un neurone de sortie.

Dans ce chapitre, nous allons étudier la faisabilité d'une mémoire synaptique à base de dispositifs memristifs. Pour cela, nous avons développé un modèle de réseau de neurones impulsionnels ainsi qu'un modèle de CBRAM en VHDL-AMS. Dans un premier temps, nous présentons le modèle de la CBRAM. Ensuite nous allons étudier différentes structures synaptiques à base de CBRAM. Enfin, nous allons vérifier la fonctionnalité du réseau à base de ces dispositifs en simulant un réseau complet en

VHDL-AMS sur une application de détection de corrélations.

Ce chapitre présente les travaux publiés dans la conférence NanoArch (International Symposium on Nanoscale Architectures) 2014 (Roclin et al. (2014)).

4.2 Modélisation VHDL-AMS d'une cellule CBRAM

4.2.1 Modèle CBRAM

Le dispositif memristif modélisé dans nos travaux appartient à la famille des Conductive-Bridge RAM (CBRAM) (Reyboz et al. (2013)). La figure 4.3 présente une image du dispositif obtenu grâce à un Microscope Electronique en Transmission (TEM). Le dispositif est composé d'une couche de GeS_2 (Chalcogénure) entre deux électrodes de métal. Cette technologie est non-volatile (Jameson et al. (2013)) et bipolaire : une tension positive aux bornes du dispositif réalise un SET et commute le dispositif dans un état de faible résistance (Low Resistance State (LRS)), alors qu'une tension négative aux bornes du dispositif réalise un RESET et commute le dispositif dans un état de haute résistance (High Resistance State (HRS)).

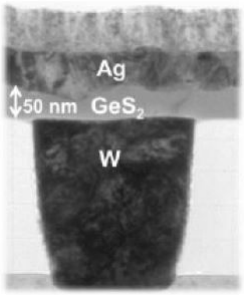


FIGURE 4.3 – Image du dispositif obtenue grâce à un Microscope Electronique en Transmission (TEM). Le dispositif est composé d'une couche de GeS_2 (Chalcogénure) entre deux électrodes de métal, une électrode active d'argent riche en ions Ag+ et une électrode passive en tungstène (W). Image du CEA LETI.

Durant un SET, quand une tension positive est appliquée sur les électrodes du dispositif, des ions Ag+ présents dans l'électrode active d'argent (anode) sont réduits par les électrons présents dans l'électrode inerte de tungstène (cathode) et créent un filament conducteur (CF) qui diminue la résistance du dispositif (LRS). Durant un RESET, quand une tension négative est appliquée sur les électrodes du dispositif, les ions Ag+ migrent, ce qui détruit le filament conducteur et augmente la résistance du dispositif (HRS).

Cette technologie n'est pas intrinsèquement multi-niveaux. Néanmoins, en utilisant une limitation de courant pendant un SET, il est possible de contrôler l'état final de faible résistance (LRS) et de réaliser un SET gradué (Yu and Wong (2010)). La valeur de haute résistance (HRS) est plus difficile à contrôler, à cause de la nature abrupte de la transition SET à RESET (Suri et al. (2013b)).

$$\frac{dh}{dt} = v_h . \exp(\frac{-E_\alpha}{kT}) . \sinh(Zq.E.\frac{\alpha}{2kT}) \tag{4.1a}$$

$$\frac{dr}{dr} = v_r . \exp(\frac{-E_\alpha}{kT}) . \sinh(\frac{\beta qV}{kT}) \tag{4.1b}$$

Pour simuler la technologie CBRAM, nous nous sommes basés sur un modèle physique du dispositif (Yu and Wong (2011)), développé en VHDL-AMS et simulé avec Spectre 7.1.1. Pour prendre en compte la limitation de courant intrinsèquement dans le

TABLE 4.1 – Paramètres physiques du modèle VHDL-AMS de la CBRAM.

Vh	50 m/s	Vitesse de formation verticale
Vr	1.5 m/s	Vitesse de formation latérale
ρ_{on}	$2.3.10^{-6}\ \Omega \times m$	Résistivité de la couche de GeS2
ρ_{off}	$8.10^{3}\ \Omega \times m$	Résistivité du filament conducteur riche en Ag
α	0.4	Dépendance aux champs électriques verticaux
β	0.35	Dépendance aux champs électriques latéraux
Δ	0.15	Seuil en dessous duquel la nucléation au niveau de l'électrode inerte est proche de zéro
Ea	0.4eV	Énergie d'activation
Ron	4,500 Ω	Résistance LRS
Roff	$10^{7}\ \Omega$	Résistance HRS

modèle, nous avons modifié le modèle en intégrant une continuité dans le calcul de la résistance équivalente du dispositif. Le modèle consiste en trois équations. La première équation 4.1a décrit la croissance verticale du filament conducteur (h) lorsque que le dispositif est à l'état OFF et la seconde équation 4.1b décrit la croissance latérale du rayon du filament (r) quand le dispositif est à l'état ON. La dernière équation représente la résistance équivalente *Rcb* de la cellule CBRAM en fonction de (h) et de (r) (équation 4.2 et figure 4.4), où L est la hauteur maximale du filament, A est l'aire totale de la CBRAM et r_{max} est le rayon maximal du filament ($A = \pi {r_{max}}^2$). Les différents paramètres du modèle (tableau 4.1) ont été ajustés grâce à la caractérisation de ces dispositifs au CEA LETI.

$$Rcb = \frac{h}{\frac{\pi.r^2}{\rho_{on}} + \frac{\pi.(r_{max}^2 - r^2)}{\rho_{off}}} + \frac{\rho_{off}.(L-h)}{A} \tag{4.2}$$

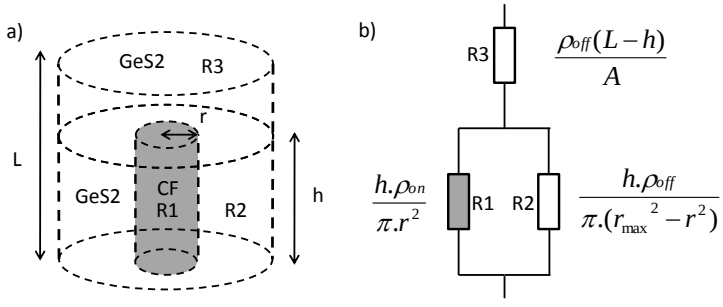


FIGURE 4.4 – a) Représentation d'un dispositif CBRAM lors de la formation d'un filament conducteur. b) Modèle de la résistance équivalente. La résistance de la couche de GeS_2 au-dessus du filament est en série avec les résistances du filament conducteur et du GeS_2 autour du filament.

La cellule CBRAM simulée dans cette étude a une résistance à l'état OFF (HRS) d'environ 10 MΩ et une résistance à l'état ON (LRS) d'environ 4,5 KΩ (résistance minimale, sans limitation de courant). Pour ajuster la résistance finale LRS, nous devons pouvoir contrôler le courant maximal circulant dans la cellule CBRAM. Pour cela, il est

possible d'ajouter un transistor en série avec la cellule CBRAM. Cette implémentation, appelée 1T1R, permet, en adaptant la tension appliquée sur la grille, de limiter le courant traversant le dispositif memristif. Nous avons simulé un dispositif CBRAM avec une limitation de courant, les résultats sont présentés figures 4.5 et 4.12. Avec cette implémentation (1T1R), la résistance à LRS de la cellule CBRAM peut varier de 4,5 KΩ à 100 KΩ avec une limitation de courant entre 1 μA et 5 μA.

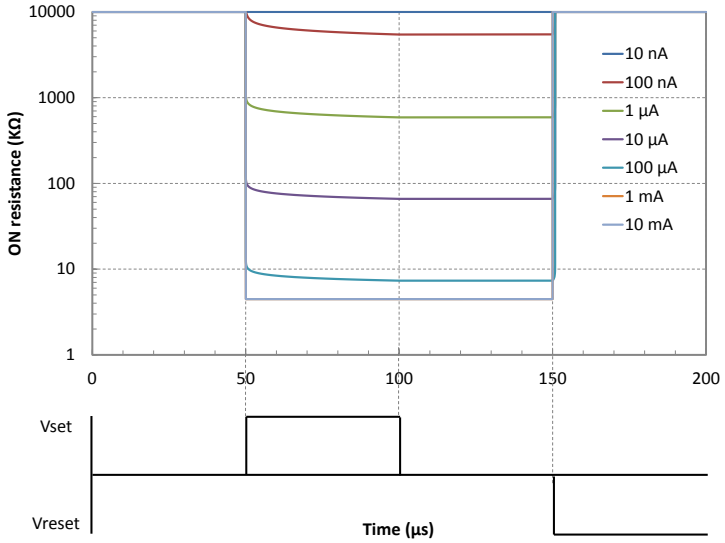


FIGURE 4.5 – Simulation d'un SET sur un dispositif CBRAM avec plusieurs valeurs de limitation de courant. La résistance LRS de la cellule CBRAM peut varier de 4,5 KΩ à 100 KΩ avec une limitation de courant entre 1 μA et 5 μA.

4.2.2 Mémoire synaptique à base de dispositifs memristifs CBRAM

Dans cette étude, nous souhaitons implémenter les synapses d'un réseau de neurones impulsionnels, de type « feedforward », avec des dispositifs CBRAM (figure 4.1). Dans ces réseaux, un potentiel d'action est propagé d'un neurone d'entrée vers un neurone de sortie. Quand ce potentiel traverse la synapse reliant ces deux neurones, il est pondéré par la valeur du poids synaptique puis est intégré par le neurone de sortie. Quand l'intégration atteint le seuil du neurone, celui-ci émet à son tour un potentiel d'action vers la couche suivante (pour laquelle il est un neurone d'entrée) et active le mécanisme d'apprentissage.

De manière analogue à l'efficacité synaptique d'une synapse, la conductance de la CBRAM peut être modifiée, conductance qui représente donc le poids synaptique. Le dispositif CBRAM est un candidat envisageable pour implémenter des synapses (Suri et al. (2013b)). Bien que la programmation de l'état LRS de la CBRAM puisse être multi-niveaux avec une limitation de courant, il est difficile de les utiliser en tant que tels, car il est difficile de contrôler les niveaux de manière fiable et précise. D'autre part, avec une méthode d'apprentissage en ligne, cela nécessite un cycle de lecture et d'écriture lors de chaque mise à jour synaptique. En effet, il faut lire l'état de la synapse, puis ajuster la limitation de courant en fonction de l'état de la précédente conductivité, et enfin reprogrammer la cellule CBRAM. Une approche plus simple est d'utiliser les dispositifs CBRAM de façon binaire ou unaire et d'implémenter plusieurs dispositifs par synapse. Dans le chapitre, «Optimisation d'un réseau de neurones impulsionnels», nous avons démontré qu'une résolution synaptique de 4 bits (ou 16 niveaux logiques) est suffisante pour implémenter une application de détection de véhicule.

En utilisant un codage binaire, comme dans une architecture de mémoire conven-

tionnelle, une mise à jour du poids synaptique exige la lecture de celui-ci afin de pouvoir l'incrémenter ou la décrémenter. Avec des dispositifs memristifs, il n'est donc pas envisageable d'utiliser le codage binaire pour des raisons de consommation d'énergie et de temps d'exécution et supprimerait le parallélisme recherché du réseau. Nous allons donc utiliser un codage unaire non-ordonné. Ce codage nous abstient du cycle de lecture de l'état des synapses, mais implique d'implémenter autant de CBRAM que de niveaux nécessaires ainsi qu'une technique de programmation stochastique.

En effet, avec un codage unaire, sans programmation stochastique, les dispositifs implémentant une synapse ne reflètent que le dernier cycle de programmation. Si une synapse est dans la fenêtre LTP, alors tous ces dispositifs commutent à '1', sinon tous à '0'. Avec une technique de programmation stochastique, chaque dispositif possède une probabilité de commutation indépendante. Lorsqu'une synapse est dans la fenêtre LTP, chaque dispositif possède une probabilité P_{LTP} de commuter à '1'. À l'inverse, quand une synapse est dans la fenêtre LTD, chaque dispositif possède une probabilité P_{LTD} de commuter à '0'. La programmation stochastique permet de mettre en place un apprentissage sur le long terme avec un codage unaire.

Cette méthode de programmation peut être implémentée en utilisant la probabilité de commutation intrinsèque au dispositif. En utilisant des impulsions proches-programmation, impulsions avec une largeur ou une amplitude plus faible, il est possible de contrôler la probabilité de commutation (LRS -> HRS et HSR -> LRS). Cela implique de connaitre, de façon reproductible, les conditions de programmation précise pour chaque probabilité de commutation. Les dispositifs actuels ne permettent pas de créer ces conditions de façon reproductible. Nous allons donc implémenter la stochasticité de manière extrinsèque aux dispositifs en utilisant des générateurs de nombres pseudo aléatoires pondérés (WPRNG) (figure 4.6).

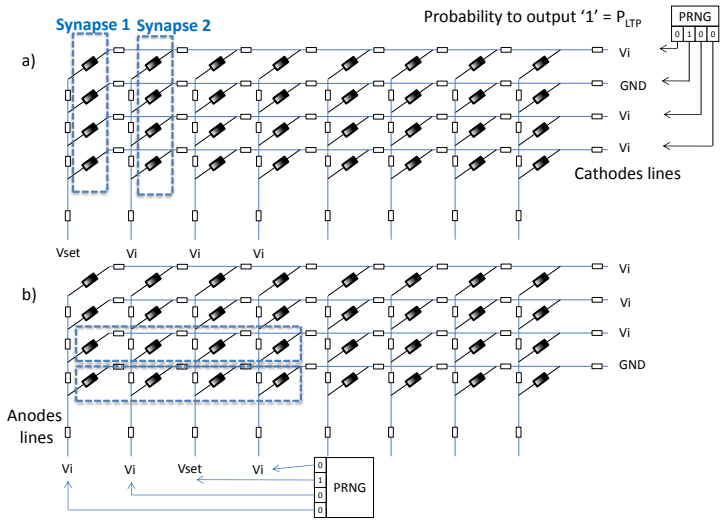


FIGURE 4.6 – Méthodes de programmation stochastique extrinsèque de deux structures synaptiques utilisant un WPRNG et 4 CBRAM pour chaque synapse.

4.3 Structure de CBRAM en crosspoint

Les technologies memristives émergentes ont l'avantage de pouvoir être arrangées d'une manière qui leur confère une grande densité d'intégration. En effet, en implémentant plusieurs lignes parallèles d'anode perpendiculaires à plusieurs parallèles lignes de cathode, il est alors possible d'insérer des cellules memristives à chaque croisement (crosspoint) (figure 4.7). Dans notre étude, nous appellerons cette topologie un crossbar.

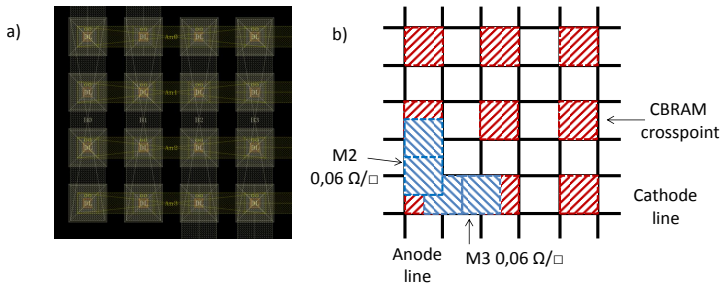


FIGURE 4.7 – a) Layout d'un crossbar 4×4 de dispositifs CBRAM co-intégrés avec un procédé CMOS 130 nm. b) Représentation de ce crossbar pour identifier les valeurs des résistances parasites, Rmesh, des couches de métaux. Rmesh est équivalent à 2□ (M2, M3 ou M4) ou 0.12Ω.

Néanmoins, la topologie crossbar soulève un problème important : les courants de fuites (sneak paths). Comme nous allons le mettre en évidence dans cette étude, les courants de fuite augmentent la consommation globale de la mémoire synaptique. Associés aux résistances parasites, les courants de fuite vont aussi limiter la taille maximale de la mémoire synaptique. Afin de réduire ces courants, il est possible d'intégrer un transistor d'accès en série avec chaque dispositif memristif. Dans notre étude, nous allons appeler cette topologie une matrice.

Dans cette étude, nous allons simuler ces deux topologies, crossbar et matrice pour pouvoir les comparer. Cette étude est primordiale pour la conception de puce neuromorphique à base de technologie memristive.

Nos simulations comprennent le modèle VHDL-AMS de CBRAM, les résistances parasites des anodes et cathodes (Rmesh) ainsi que les résistances parasites des lignes d'accès aux anodes/cathodes (Raccess) comme l'illustre la figure 4.7. Nous appelons anode, chaque ligne connectant les anodes des CBRAM, et nous appelons cathode, chaque ligne connectant les cathodes des CBRAM (figures 4.7 et 4.6). Pour effectuer des simulations avec des valeurs de résistances parasites proches de la réalité, nous avons réalisé le layout d'un crossbar de CBRAM avec un procédé CMOS de 130 nm pour en extraire les valeurs de résistances. Ce procédé CMOS a une résistance surfacique de 0.06Ω/□ ainsi qu'une largeur minimum d'interconnections de métal de 0.2μm (pour M2 and M3). Dans nos simulations, nous avons estimé la valeur de Rmesh à 0.12Ω et Raccess à 5Ω (ce qui correspond à une ligne de métal (M2 ou M3) de 0.2μm×16μm) (figure 4.7).

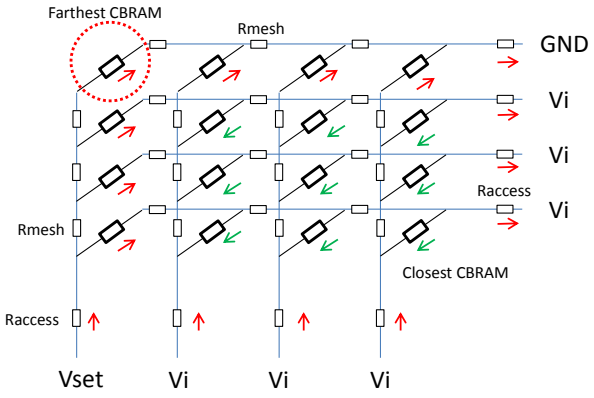
Pour les cellules CBRAM, le cycle de programmation RESET est symétrique au cycle de programmation SET et est moins contraignant (largeur d'impulsion plus courte pour une même tension). Nous allons donc, dans cette étude, nous focaliser sur le cycle qui va limiter les topologies étudiées, le cycle de programmation SET.

4.3.1 Crossbar

Le crossbar est la topologie qui offre le maximum de densité d'intégration (sans transistor de sélection). Pour ne pas induire la commutation parasite des cellules voisines (cellules partageant une anode ou une cathode avec la cellule en programmation) lors de la programmation d'une cellule CBRAM, nous devons introduire une tension intermédiaire Vi égale à la moitié de la tension $Vset$.

Courant de fuite

Pour mettre en évidence les courants de fuite lors de la programmation d'une cellule CBRAM, nous avons reproduit le pire cas de programmation en imposant un état de



331.41 µA	165.52 µA	165.53 µA	165.55 µA	CA3	828.01 µA
165.52 µA	0.38 µA	0.37 µA	0.37 µA	CA2	164.38 µA
165.53 µA	0.37 µA	0.37 µA	0.36 µA	CA1	164.41 µA
165.55 µA	0.37 µA	0.36 µA	0.36 µA	CA0	164.44 µA
AN3	AN2	AN1	AN0		
828.01 µA	164.38 µA	164.41 µA	164.44 µA		

FIGURE 4.8 – Structure crossbar – Condition de programmation SET – Valeurs des courants circulant dans le crossbar pendant le SET de la cellule CBRAM la plus éloignée des circuits d'écriture. Toutes les CBRAM sont dans l'état LRS à 4,5KΩ (pire cas). Les courants de fuites sont les plus importants dans les cellules voisines (cellules partageant une anode ou une cathode avec la cellule en programmation) et qui ont la tension Vi à leurs bornes.

haute conductance LRS (4,5K Ω) pour toutes les CBRAM du crossbar. La figure 4.8 présente les conditions de programmation d'un SET sur la cellule la plus éloignée des circuits d'écriture dans un crossbar (4×4) ainsi qu'un instantané des courants circulant dans chaque CBRAM et transistor d'accès. On peut voir que $Vset$ est appliqué sur l'anode et GND sur la cathode de la cellule sélectionnée. Toutes les autres anodes et cathodes reçoivent Vi afin de ne pas induire la commutation parasite des cellules voisines. Cette figure montre qu'un courant total de 1.32 mA est consommé lors d'un SET alors qu'un courant de 331 μA était suffisant pour faire commuter la cellule. Les courants de fuite sont majoritairement présents sur les cellules voisines, c'est-à-dire, sur les cellules partageant une anode ou cathode avec la cellule sélectionnée pour le SET et ayant la tension Vi à leurs bornes. En ne prenant en compte que ces courants de fuite, on peut simplifier le calcul de la consommation en courant de fuite par $(N-1)i_{set}$ dans le pire cas ou $(N_{ON}-1)\frac{i_{set}}{2}$ dans tous les autres cas, où N_{ON} est le nombre de cellules CBRAM dans un état de haute conductance partageant une anode ou cathode avec la cellule sélectionnée pour le SET.

Chutes de tension

Les chutes de tension sur les lignes d'anodes ou de cathodes sont dues aux résistances parasites ainsi qu'aux courants de fuites. Bien qu'une CBRAM soit capable de commuter complètement avec des durées d'impulsion inférieure à la microseconde, avec $Vset$=1,5 V, il est nécessaire d'augmenter la durée des impulsions pour permettre la programmation avec des tensions inférieures dues aux chutes de tension. La chute de tension maximale se produit au niveau des dispositifs les plus éloignés des circuits d'écriture. Elle détermine la taille maximale du crossbar. En effet, cette chute de tension doit être suffisamment faible pour que la tension effective aux bornes du dispositif soit supérieure à la tension minimale de programmation d'un SET pour une durée d'impulsion donnée. Pour trouver cette valeur de tension de SET minimale, nous avons effectué plusieurs SET sur une cellule CBRAM avec des tensions de SET de 1.1 V à

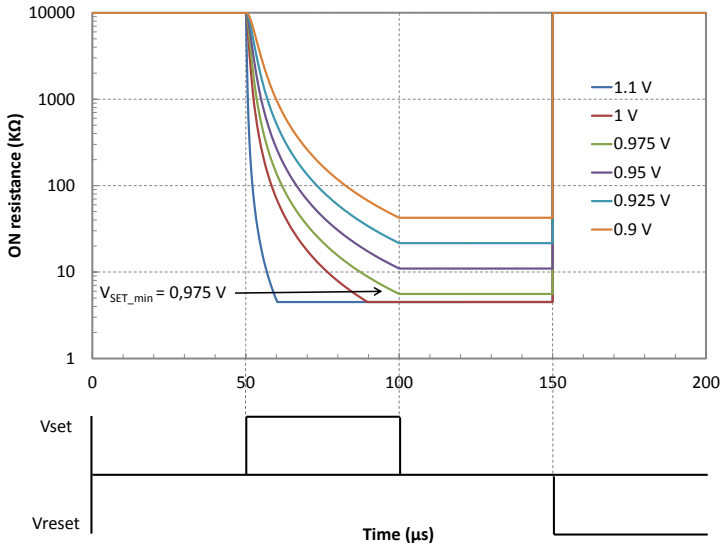


FIGURE 4.9 – Évolution de la résistance LRS d'une cellule CBRAM après programmation avec des tensions inférieures à V_{SET}. La cellule CBRAM n'effectue pas un SET complet pour des tensions inférieures à $V_{SET\_min} = 0.975V$ avec une durée d'impulsion de 50μs.

0.9 V pour une impulsion de 50μs. Les résultats obtenus sont présentés figure 4.9. Nous pouvons voir qu'une cellule CBRAM n'effectue pas un SET complet avec des valeurs de tension de programmation inférieure à V_{SET_min}=0.975 V . La tension V_{SET_min} correspond donc à la tension minimale de programmation autorisée aux bornes d'un dispositif dans notre crossbar.

À cause de limitation de mémoire, notre simulateur n'est pas capable de simuler des crossbars dont la taille est supérieure à 125×125. Pour calculer la chute de tension pour des crossbars de plus grandes dimensions, nous avons simplifié le crossbar en éliminant les chemins où les courants ne sont pas significatifs grâce aux résultats de la simulation de la figure 4.8. La figure 4.10 a) illustre le schéma résultant de cette simplification. En utilisant des transformations $Y - \triangle$ récurrentes (figure 4.10 c)), il est possible de continuer la simplification du crossbar jusqu'à une topologie finale en Y avec des résistances équivalentes $R1eq$, $R2eq$ et $R3eq$ (figure 4.10 b)). La valeur effective de tension de programmation est alors calculée grâce au théorème de Millman.

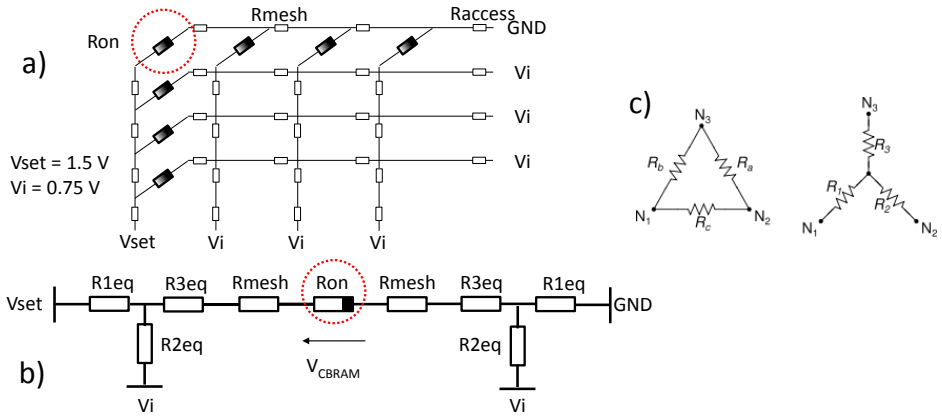


FIGURE 4.10 – a) Représentation du crossbar pendant un SET dont les chemins où le courant est négligeable sont éliminés. b) Modélisation du crossbar avec des résistances équivalentes après simplification. c) Transformation $Y - \triangle$ utilisée.

Le script, développé en Python, permettant de calculer la tension effective de SET

lors de la programmation de la cellule CBRAM la plus éloignée dans le pire cas, est présenté dans le listing 4.1.

Listing 4.1 – Code Python permettant le calcul de la tension effective de SET lors de la programmation de la cellule CBRAM la plus éloignée dans le pire cas.

```
#! /usr/bin/env python
#init des valeurs de resistance et des tensions
Rmesh=0.12
Ron=4500.0
Raccess=5.0
Vset=1.5
Vi=0.75
#init taille du crossbar (size x size)
size=1000
#init de la valeur de resistance equivalente du crossbar
Req=Raccess
#init des valeurs Ra Rb Rc de la transformation Y-Delta
Ra=Ron
Rb=Ron
Rc=Rmesh
#debut de la boucle de Y-Delta
for cycle in xrange(1,size-1):
                #calcul des resistances equivalente Y-Delta
                R1=(Rb*Rc)/(Ra+Rb+Rc)
                R2=(Ra*Rc)/(Ra+Rb+Rc)
                R3=(Ra*Rb)/(Ra+Rb+Rc)
                #mise a jour de la valeur de resistance equivalente du crossbar
                Req=Req+R1
                #mise a jour des valeurs Ra Rb Rc
                Ra=Ron
                Rb=R3
                Rc=R2+Rmesh
#calcul de la tension effective apres simplification du reseau
R1=Req
R2=R3
alpha=(1/R1)+(1/Ron)+(1/R2)
Vb=((Vset*Ron*alpha/R1)+(Vi/R2)+(Vi*Ron*alpha/R2))/((Ron*alpha*alpha)-(1/Ron))
Vc=((Vb/Ron)+(Vi/R2))/alpha
Veff=Vb-Vc
#affichage de la tension effective
print "Veff=%s"%(Veff)
```

Les résultats des simulations VHDL-AMS (pour les crossbars de taille inférieure à 125×125) et des simulations Python (pour les crossbars de taille supérieure à 125×125) sont présentés figure 4.11. Ces résultats montrent qu'avec une résistance LRS de 4,5 KΩ, la tension minimale V_{SET_min} est atteinte pour un crossbar de taille 150×150, ce qui correspond à dix synapses si l'on considère 16 CBRAM pour chaque synapse (16 niveaux logiques).

Pour augmenter la taille maximale d'un crossbar, il faut diminuer l'un des deux éléments qui induisent des chutes de tension, des résistances parasites ou des courants de fuite. La réduction des résistances parasites n'est pas envisageable, car elles sont intrinsèques à un procédé CMOS et à une technologie mémoire donnée (excepté en augmentant la largeur des connexions ce qui altère la densité), il faut donc se pencher sur les courants de fuite. En augmentant la valeur de la résistance LRS, il est possible de diminuer les courants circulant dans chaque cellule pour une tension donnée et donc, les courants de fuites. Pour augmenter la résistance LRS, il faut, comme énoncé précédemment, diminuer le courant circulant dans une CBRAM lors de sa programmation dans l'état LRS. Figure 4.12 montre les différents niveaux de résistance LRS atteignables en contrôlant le courant de programmation. Il est possible, avec cette technologie d'at-

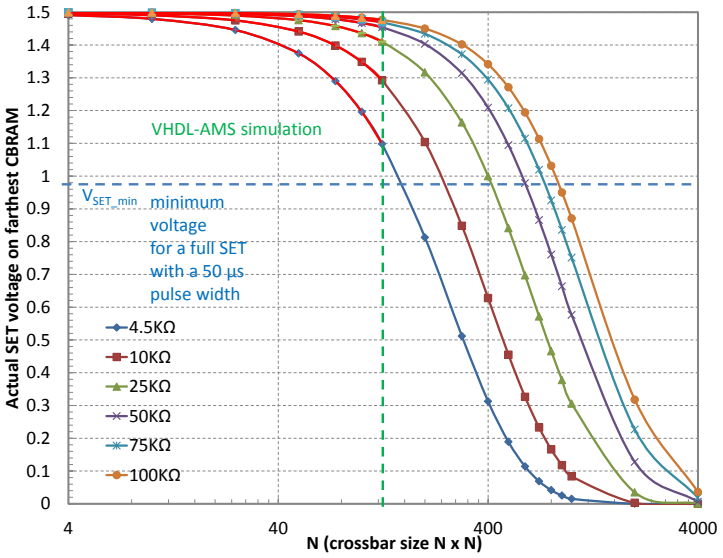


FIGURE 4.11 – Évolution de la tension effective de SET de la CBRAM la plus éloignée avec des valeurs de résistances LRS de 4,5 KΩ à 100KΩ pour des crossbars de taille 4×4 à 4000×4000. Nous avons superposé les résultats de simulation VHDL-AMS et Python pour les crossbars de tailles inférieures à 125×125. Avec une résistance LSR de 4,5 KΩ, la tension minimale V_{SET_min} est atteinte pour un crossbar de taille 150 × 150. Avec une résistance LSR de 100 KΩ, la tension minimale V_{SET_min} est atteinte pour un crossbar de taille 150 × 150. De plus larges crossbars peuvent être implémentés en utilisant des valeurs de résistance LRS supérieure.

teindre des niveaux de résistance situés entre 4,5 KΩ et 100 KΩ, avec des courants maximaux de programmation située entre 1 μA et 5 μA. Malheureusement, en topologie crossbar, il n'y a pas d'éléments extérieurs pouvant réduire ou contrôler le niveau de courant circulant dans une CBRAM, il n'est donc pas possible d'ajuster le niveau de résistance LRS final. Nous allons quand même effectuer des simulations de crossbar avec de plus grandes valeurs de résistance LRS pour en tirer les conséquences. Cela pourra, en fonction des résultats, inciter les fondeurs de mémoire émergente à modifier les couches de matériaux pour atteindre un niveau de résistance LRS supérieur.

Figure 4.11 montre qu'en augmentant le niveau de résistance LRS, il est possible d'implémenter des crossbars de taille plus significative, par exemple avec une résistance LRS de 100 KΩ il est alors possible d'atteindre des crossbars de tailles 875 × 875 (765 625 cellules CBRAM ou 47 851 synapses (de 16 niveaux).

Variation du courant de lecture

Les chutes de tension affectent aussi le courant de lecture. En effet, en fonction de l'emplacement de la cellule CBRAM lu dans le crossbar et des états de ses cellules voisines (LRS ou HRS), le courant de lecture ne sera pas identique pour un même état. La figure 4.13 montre la différence de courant lors de la lecture entre la CBRAM la plus proche et la CBRAM la plus éloignée des circuits d'écriture quand toutes les cellules CBRAM sont à l'état LRS (pire cas). On remarque qu'avec une résistance LRS de 4,5 KΩ, une différence de 10% est atteinte pour un crossbar de taille 70 × 70. De plus fortes valeurs de résistances LRS permettent de diminuer cette différence : avec une résistance LRS de 50 KΩ, la différence se limite à 5% pour un crossbar de taille 150 × 150.

Dans un réseau de neurones impulsionnels, le courant de lecture est intégré par les neurones de sortie. Une différence dans ces courants, provoquée par leur emplacement dans le crossbar et par l'état (LRS ou HRS) des cellules environnantes, peut conduire à un biais lors de leur intégration. L'intégration au niveau des neurones de sortie peut

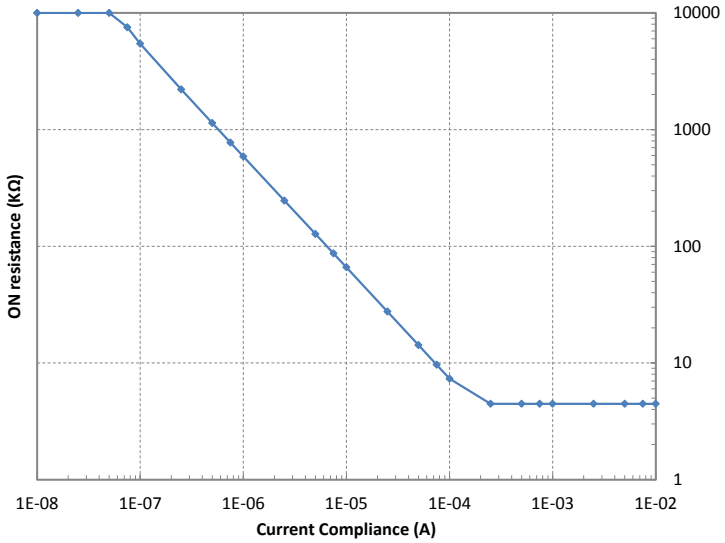


FIGURE 4.12 – Évolution de la valeur de résistance LRS en fonction du courant de programmation. Des niveaux de résistance LRS situés entre 4.5 KΩ à 100 KΩ peuvent être atteints avec des courants de programmation située entre 1 μA et 5 μA.

s'implémenter de deux manières différentes : analogique ou numérique. Avec une intégration numérique, le courant est comparé à une valeur seuil pour différencier l'état des CBRAM lues, LRS ou HRS ('0' ou '1'). La valeur unaire obtenue est ajoutée à la valeur de l'intégrateur grâce à un accumulateur numérique. Dans ce cas, la différence dans la valeur lue n'est pas problématique tant que la marge de lecture des CBRAM reste assez élevée pour pouvoir proprement discerner les deux états que peut prendre une CBRAM. Avec une intégration analogique, le courant de lecture est directement intégré par un condensateur dans le neurone de sortie. Le courant de lecture de deux cellules CBRAM dans le même état a donc besoin d'être identique et non pas dépendant de la localisation des CBRAM ou de l'état de leurs cellules voisines. Dans ce cas, la différence de courant de lecture peut avoir un impact sur le fonctionnement global et les performances d'un réseau de neurones impulsionnels. Dans le cas d'un apprentissage non-supervisé, il est possible que celui-ci puisse compenser cette disparité, c'est pourquoi ce sujet est dans nos perspectives de travaux futurs.

Perturbation des cellules voisines

Une autre problématique liée au crossbar est la commutation non désirée des cellules voisines. En effet, on peut voir sur la figure 4.8 que les cellules CBRAM partageant une anode ou une cathode avec la cellule en train d'être programmée possèdent la tension Vi à leurs bornes, ce qui est trop juste pour faire commuter ces CBRAM avec une impulsion de $50\mu s$ mais initie déjà sa commutation. Nous avons voulu connaitre la durée d'impulsion requise pour faire commuter une cellule CBRAM complètement avec Vi comme tension de SET. La figure 4.14 montre les durées d'impulsion requises pour faire commuter totalement une cellule CBRAM en fonction de la valeur de la tension de SET (avec une résistance LRS de 4,5 KΩ). On peut voir qu'avec Vi en tension de SET, la cellule CBRAM a besoin d'une impulsion de programmation de 1,12 ms pour pouvoir commuter de façon complète. Cela signifie que si une cellule CBRAM est programmée 22 fois de suite avec des impulsions de $50\mu s$ ($22 \times 50\mu s = 1,12ms$), alors toutes les cellules CBRAM partageant une anode ou une cathode avec cette cellule risque de subir une commutation complète non désirée. Pour augmenter le nombre d'impulsions requis pour faire commuter les cellules voisines, il est possible de diminuer

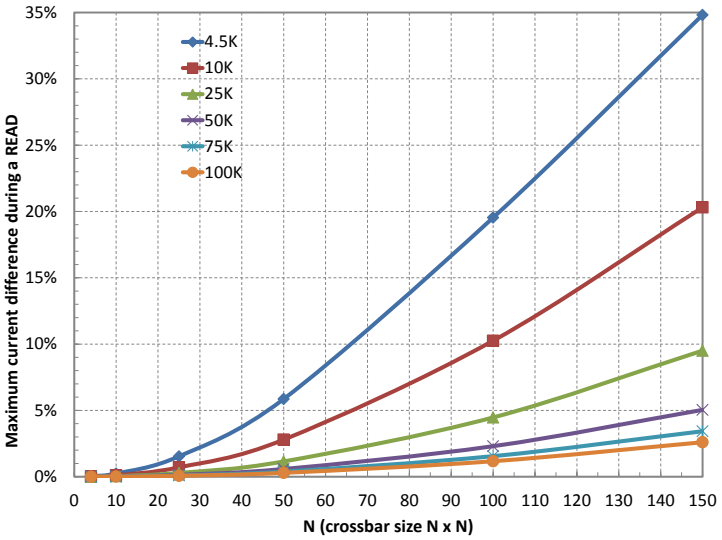


FIGURE 4.13 – Différence de courant de lecture entre la CBRAM la plus proche et la plus éloignée dans le pire cas (toutes les CBRAM dans l'état LRS) pour différentes tailles de crossbar et résistance LRS.

la durée d'impulsion de programmation, mais cela augmenterait la valeur V_{SET_min} (figure 4.14) et donc limiterait la taille maximale du crossbar. Il faut donc trouver un compromis entre la durée d'impulsion de programmation, la valeur V_{SET_min}, et la taille maximale du crossbar.

4.3.2 Matrice connectée par les anodes

Contrairement aux crossbars, les matrices possèdent un transistor en série avec chaque cellule memristive. Ce transistor permet de sélectionner la cellule qui doit être programmée afin d'éliminer les courants de fuite. Pour préserver les hautes densités d'intégration qu'offrent ces technologies, il n'est pas possible d'adresser individuellement les cellules, car il faudrait autant de bits de contrôle et de lignes d'accès que de cellules CBRAM. Nous allons plutôt connecter toutes les grilles des transistors ensemble en suivant soit les lignes des anodes soit les lignes des cathodes (figure 4.15).

Ces deux structures sont symétriques et ont des avantages complémentaires. D'un côté, la matrice connectée par les anodes permet, en mode lecture, de sélectionner uniquement les synapses appartenant à un même neurone d'entrée ce qui réduit la différence des courants de lecture des CBRAM. D'un autre côté, la matrice connectée par les cathodes permet, en mode de programmation, de sélectionner uniquement les synapses appartenant à un même neurone de sortie, ce qui permet de limiter les courants de fuite et donc de maximiser la taille de la matrice. À l'inverse, la matrice connectée par les anodes en mode programmation requiert de sélectionner toutes les cellules CBRAM et donc de retrouver une topologie crossbar (en considérant une programmation parallèle, car une programmation séquentielle, une synapse après l'autre, supprime le parallélisme naturel des réseaux impulsionnels). De même, la matrice connectée par les cathodes en mode lecture requiert la sélection de toutes les CBRAM et donc une topologie crossbar.

Courant de fuite

Les conditions de programmation de la matrice connectée par les anodes sont affichées figure 4.16. Tout d'abord, on peut remarquer que la tension intermédiaire Vi n'est plus nécessaire si les synapses sont programmées séquentiellement. Ensuite, les

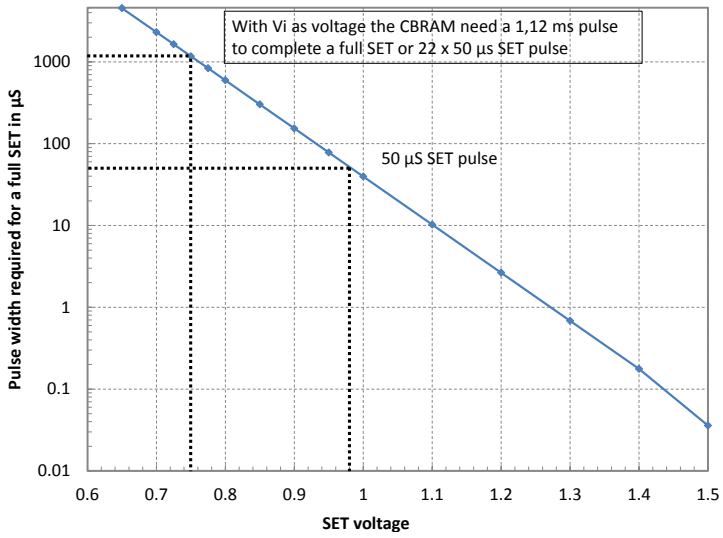


FIGURE 4.14 – Temps d'impulsion requis avec plusieurs tensions de SET pour effectuer une commutation complète de SET. Avec la tension Vi, une impulsion de 1.12 ms est requise soit 22 impulsions de $50\mu s$.

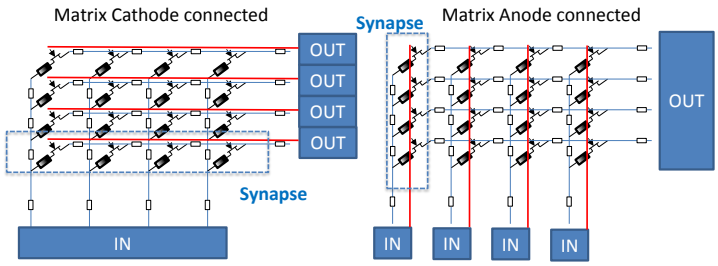


FIGURE 4.15 – 2 structures en matrice : matrice connectée par la cathode et matrice connectée par l'anode.

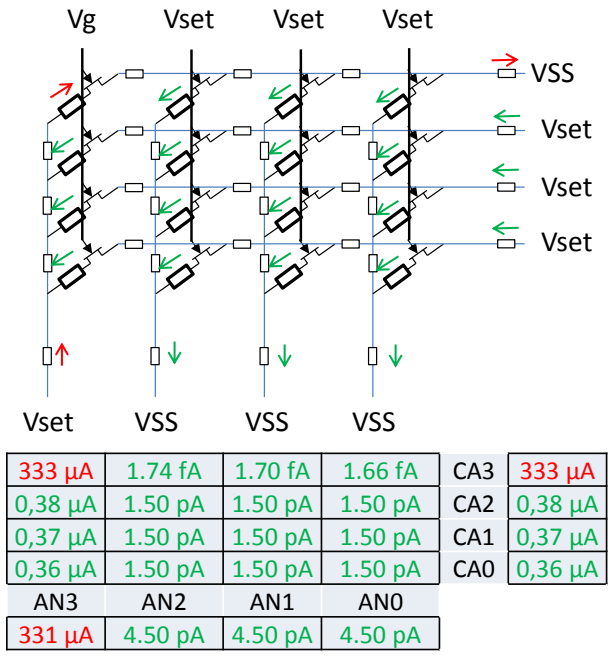
courants de fuite circulant dans les cellules voisines, partageant une anode ou une cathode avec la cellule en programmation, sont minimaux et ne sont dus qu'aux chutes de tension créées par les résistances parasites des lignes. Cette matrice ne consomme que les $333\mu A$ nécessaires à la commutation de la cellule CBRAM en programmation et donc $((N_{ON} - 1)\frac{i_{set}}{2})$ moins que la structure en crossbar.

Chutes de tension

Comme pour le crossbar, nous avons développé un script Python pour calculer la tension effective de programmation de la CBRAM la plus éloignée qui prend en compte les chutes de tension pour des matrices de taille supérieure à 125×125. Figure 4.17 a) présente le schéma de la matrice connectée par les anodes où les chemins dont les courants sont négligeables ont été supprimés. Puis en utilisant les mêmes méthodes algorithmiques que pour le crossbar, nous avons simplifié cette structure avec des résistances équivalentes, illustrée sur la figure 4.17 b). Le script Python est, lui, présenté dans le listing 4.2.

Listing 4.2 – Code Python permettant le calcul de la tension effective de SET lors de la programmation de la cellule CBRAM la plus éloignée dans le pire cas pour la matrice connectée par les anodes.

```
1 #! /usr/bin/env python
2 #init des valeurs de resistance et des tensions
3 Rmesh=0.12
```



333 µA	1.74 fA	1.70 fA	1.66 fA	CA3	333 µA
0,38 µA	1.50 pA	1.50 pA	1.50 pA	CA2	0,38 µA
0,37 µA	1.50 pA	1.50 pA	1.50 pA	CA1	0,37 µA
0,36 µA	1.50 pA	1.50 pA	1.50 pA	CA0	0,36 µA
AN3	AN2	AN1	AN0		
331 µA	4.50 pA	4.50 pA	4.50 pA		

FIGURE 4.16 – Matrice connectée par les anodes – Condition de programmation du SET – Courant dans chaque CBRAM et résistance d'accès pendant un SET avec toutes les CBRAM dans l'état LRS à 4,5 KΩ (pire cas).

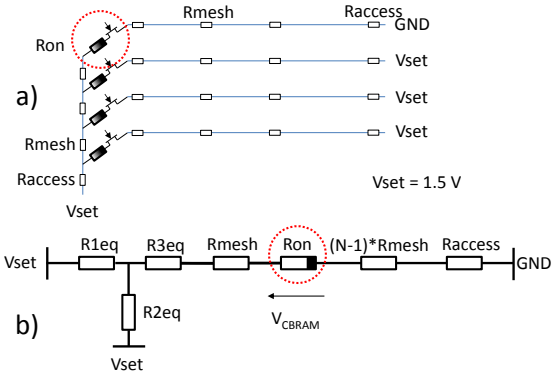


FIGURE 4.17 – a) Représentation de la matrice connectée par les anodes où les chemins dont les courants sont négligeables ont été supprimés pendant un SET avec toutes les CBRAM ON (LRS). b) Modèle de matrice avec des résistances équivalentes après simplification.

```
4  Ron=4500.0
5  Raccess=5.0
6  Vset=1.5
7  #init taille du crossbar (size x size)
8  size=1000
9  #init de la valeur de resistance equivalente du crossbar
10 Req=Raccess
11 #init des valeurs Ra Rb Rc de la transformation Y-Delta
12 Ra=Ron+((size-1)*Rmesh)+Raccess
13 Rb=Ron+((size-1)*Rmesh)+Raccess
14 Rc=Rmesh
15 #debut de la boucle de Y-Delta
16 for cycle in xrange(1,size-1):
17                 #calcul des resistances equivalente Y-Delta
18                 R1=(Rb*Rc)/(Ra+Rb+Rc)
19                 R2=(Ra*Rc)/(Ra+Rb+Rc)
20                 R3=(Ra*Rb)/(Ra+Rb+Rc)
21                 #mise e jour de la valeur de resistance equivalente du crossbar
```

```
22                Req=Req+R1
23                #mise a jour des valeurs Ra Rb Rc
24                Ra=Ron+((size-1)*Rmesh)+Raccess
25                Rb=R3
26                Rc=R2+Rmesh
27  #calcul de la tension effective apres simplification du reseau
28  Rmiddle=R2+Rmesh+Ron+((size-1)*Rmesh)+Raccess
29  R1=Req
30  R2=R3
31  Vb=((Vset/R1)+(Vset/R2))/((1/R1)+(1/R2)+(1/Rmiddle))
32  i3=(Vb)/Rmiddle
33  Vp=Vb-(Rmesh*i3)
34  Vm=Vp-(Ron*i3)
35  Veff=Vp-Vm
36  #affichage de la tension effective
37  print "Veff=%s"%(Veff)
```

La figure 4.18 montre la tension effective, due aux chutes de tension, lors de la programmation de la cellule CBRAM la plus éloignée dans le pire cas (toutes les CBRAM dans l'état de résistance LRS). Avec cette topologie, la chute de tension est plus faible qu'avec la topologie crossbar. Avec une valeur de résistance LRS de 4,5 KΩ, il est possible d'implémenter une mémoire synaptique de 20000×20000. Avec une résistance LRS de 100 KΩ, une taille supérieure à 400000 × 400000 peut être théoriquement atteinte (160 milliards de cellules CBRAM ou dix milliards de synapses avec seize CBRAM par synapse), mais d'autres éléments pourraient venir limiter la taille de la matrice (courant de lecture, yield de fabrication, variation du procédé de fabrication).

Comme énoncé précédemment, pour atteindre ce nombre de synapses avec une matrice connectée par l'anode, il faut programmer les synapses séquentiellement pour ne pas avoir à sélectionner l'ensemble des CBRAM comme dans une structure crossbar.

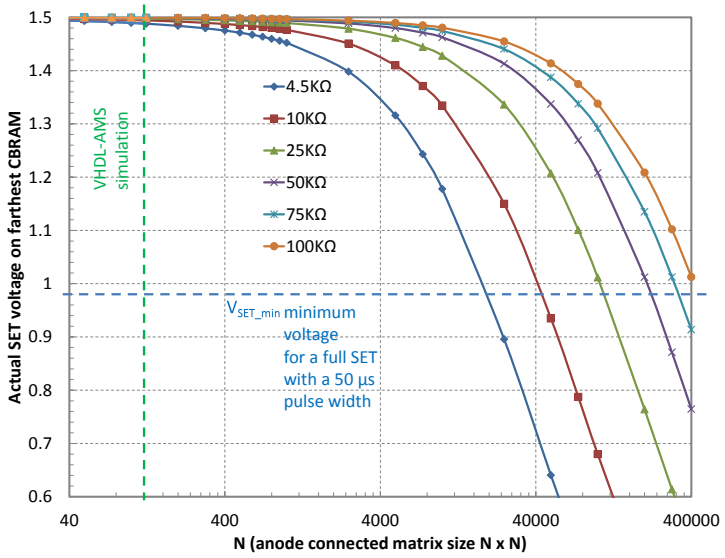


FIGURE 4.18 – Évolution de la tension effective de programmation de la CBRAM la plus éloignée pendant un SET dans le pire cas avec des résistances LRS variant de 4,5 KΩ à 100 KΩ et des tailles de matrices connectées par les anodes variant de 4×4 à 400,000×400,000.

Variation du courant de lecture

Cette structure permet en mode lecture de sélectionner uniquement les synapses d'un même neurone d'entrée. La figure 4.19 montre la différence de courant de lecture entre la CBRAM la plus proche et la plus éloignée des circuits d'écriture dans la matrice.

On peut voir qu'avec une résistance LRS de 4,5 KΩ, une différence de courant de 10 % apparait à partir d'une taille de matrice de 90×90 au lieu de 65×65 pour la structure en crossbar. Avec une résistance LRS de 100 KΩ, la différence est inférieure à 1% pour une taille de matrice de 100×100.

Dans un premier temps on peut remarquer que la différence des courants de lecture est élevée pour des valeurs de résistance LRS faibles. On peut aussi remarquer que la différence du crossbar tend à être le double de celle de la matrice ce qui montre l'efficacité de cette structure même si pourtant cette différence reste importante. Comme avec le crossbar, des valeurs LRS plus élevées diminuent la différence entre les courants de lecture minimum et maximum, mais génèrent des courants de lecture plus faibles. Avec un neurone numérique, cela signifie que le comparateur de courants (pour différencier un état ON d'un état OFF) doit être plus sensible. Avec un neurone analogique, ces faibles courants sont intégrés par un condensateur, ce qui demande un niveau de précision supérieur, parfois difficile à atteindre en analogique du fait de la dispersion des paramètres technologiques.

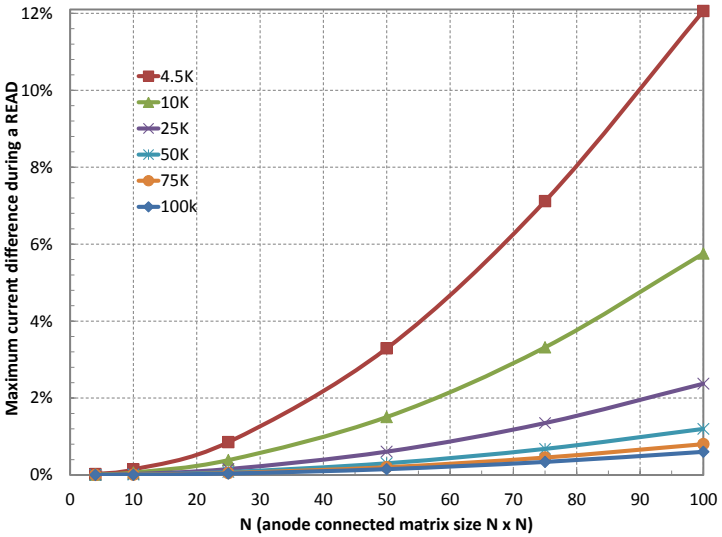


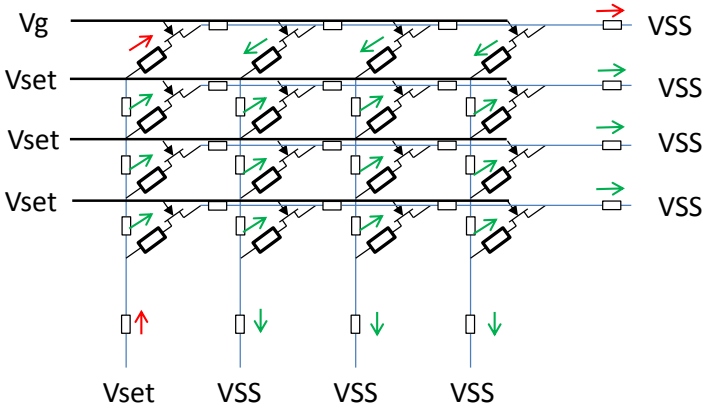
FIGURE 4.19 – Évolution de la différence des courants de lecture de la CBRAM la plus proche et la plus éloignée des circuits d'écriture dans la matrice connectée par les anodes pour des tailles de matrices et des valeurs de résistances LRS différentes.

4.3.3 Matrice connectée par les cathodes

Avec cette topologie, un neurone de sortie est capable de sélectionner uniquement l'ensemble de ses synapses afin de pouvoir les programmer en parallèle. Cependant, tous les neurones de sortie intègrent les impulsions des neurones d'entrée en mode lecture, la sélection de l'ensemble des CBRAM est alors nécessaire ce qui correspond à la topologie en crossbar.

Courant de fuite

Les conditions de programmation de la matrice connectée par les cathodes sont présentées dans la figure 4.20. On peut remarquer tout d'abord la symétrie avec la précédente structure. Comme celle-ci, elle ne consomme que le courant nécessaire à la commutation de la CBRAM sélectionnée.



333 µA	0,38 µA	0,37 µA	0,36 µA	CA3	331 µA
1.50 pA	0	0	0	CA2	1.50 pA
1.50 pA	0	0	0	CA1	1.50 pA
1.50 pA	0	0	0	CA0	1.50 pA
AN3	AN2	AN1	AN0		
333 µA	0,38 µA	0,37 µA	0,36 µA		

FIGURE 4.20 – Matrice connectée par les cathodes – Condition de programmation SET – Courant dans chaque CBRAM et résistances d'accès pendant un SET dans le pire cas (toutes les CBRAM LRS à 4.5KΩ).

Chutes de tension et Variation du courant de lecture

Due à leur symétrie, la tension effective aux bornes de la CBRAM la plus éloignée est identique à la structure de matrice précédente (figure 4.18). Cette structure doit sélectionner toutes les CBRAM en mode lecture. La différence de courant est donc identique à celle de la structure en crossbar (figure 4.19).

4.3.4 Matrice – Surcoût associé au transistor de sélection dans une matrice

Les matrices permettent d'intégrer un plus grand nombre de dispositifs grâce à leurs transistors de sélection qui permettent d'éliminer les courants de fuite. Nous avons estiméle coût en superficie de l'ajout de ce transistor de sélection. Pour se faire, nous avons à notre disposition le layout d'un crossbar 4 et d'une matrice 4 de dispositifs CBRAM intégré dans un procédé CMOS 130nm (figure 4.21).

La superficie du crossbar 4×4 est de $5,12\mu m \times 5,12\mu m$ $(26,21\mu m^2)$, ce qui donne une superficie pour une cellule CBRAM d'un crossbar de $1,28\mu m \times 1,28\mu m$ $(1,63\mu m^2)$. Cette superficie est bien supérieure $(\times 40,63)$ à la superficie de l'empilement CBRAM qui est de $0,2\mu m \times 0,2\mu m$ $(0,04\mu m^2)$. Cela est dû aux règles de design (width + spacing) des différents vias et couches de métal de la technologie CMOS 130 nm. La superficie de la matrice 4×4 est de $5,4\mu m \times 6,92\mu m (37,37\mu m^2)$, ce qui est 42% supérieur au crossbar (figure 4.21).

4.3.5 Conclusions des études réalisées

Ces études exposent les différents problèmes associés à la lecture dans le crossbar ainsi que les commutations non-désirées des dispositifs CBRAM lors des phases de programmation, mais aussi les limites du crossbar en termes de taille et de consommation. En les associant, les deux structures en matrices ont montré leurs aptitudes à résoudre ces difficultés. Pourtant aucune d'entre elles ne permet de sélectionner soit une seule

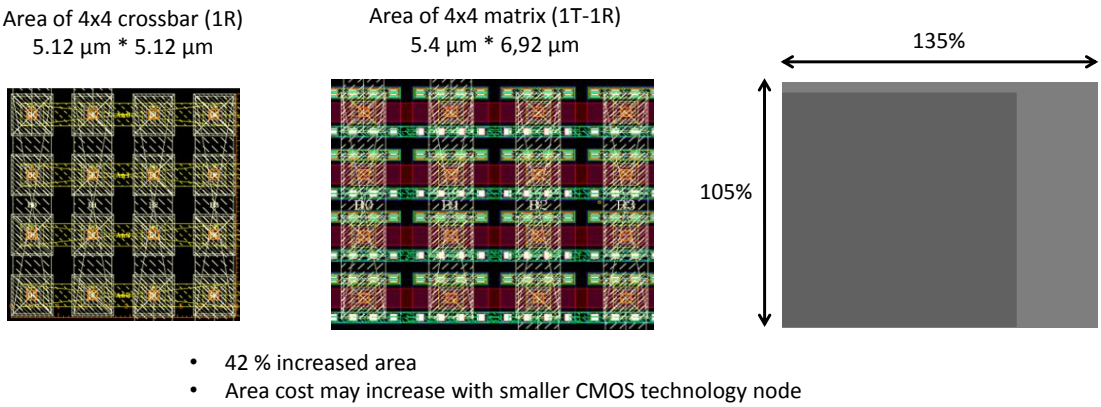


FIGURE 4.21 – a) Layout crossbar 4×4 b) Layout matrice 4×4 de dispositifs memristifs CBRAM intégrés dans un procédé CMOS 130nm. Comparaison à l'échelle de la superficie des deux structures. La matrice consomme une superficie 42% supérieur à la superficie du crossbar.

anode, soit une seule cathode. En effet la structure parfaite devrait être capable de sélectionner en mode lecture une seule anode (celle correspondant au neurone d'entrée activé) et en mode programmation une seule cathode (celle correspondant au neurone de sortie activé). Il est difficile d'atteindre ce but sans accroitre davantage la surface de silicium utilisée. Une solution serait d'ajouter un second transistor en série. Un de ces deux transistors serait contrôlé par un neurone d'entrée dans les phases de lecture (ON quand ce neurone s'active) et il serait toujours en saturation pour les phases de programmation. Le second, contrôlé par le neurone de sortie dans les phases de programmations (quand ce neurone s'active) et il serait toujours en saturation dans les phases de lectures. Cette solution pourrait résoudre en une seule structure les limites du crossbar, mais le coût en surface serait important. Notre étude du layout de la matrice montre une augmentation en surface de 42 % par rapport au crossbar bien que chaque transistor de sélection soit placé au-dessous du stack CBRAM. Un ajout d'un second transistor obligatoirement à côté de l'autre multiplierait par deux la surface des cellules (1T-1T-1R) CBRAM.

Lors de la conception de champs synaptique à base de dispositifs résistifs, il faut donc faire un choix entre la structure en crossbar qui permet une densité d'intégration maximale avec un coût en taille et en consommation ou entre une des structures en matrice qui ne peuvent répondre au problème que complémentairement.

4.4 Modélisation d'un réseau de neurones impulsionnels et simulation VHDL-AMS

4.4.1 Réseau de neurones impulsionnels en VHDL-AMS

Pour valider, au niveau circuit, un réseau de neurones impulsionnels à base de dispositifs memristifs, nous avons développé un modèle VHDL-AMS de ce réseau. Ce langage de description matériel bas niveau permet notamment de décrire le comportant analogique de chaque composant du circuit et de les associer à des composants numériques classiques. Les simulations VHDL-AMS apportent un degré de précision, mais à un coût important, le temps de simulation. Nous avons donc simulé une application peu couteuse en termes de neurones ou de synapses, par exemple une application de détection de corrélations dans un motif d'entrée. Nous avons généré un motif de quatre entrées (figure 4.27) grâce à XNET, dont deux (entrées trois et quatre) sont temporellement corrélées. Avec ce motif, un réseau impulsionnel possédant un seul neurone de sortie

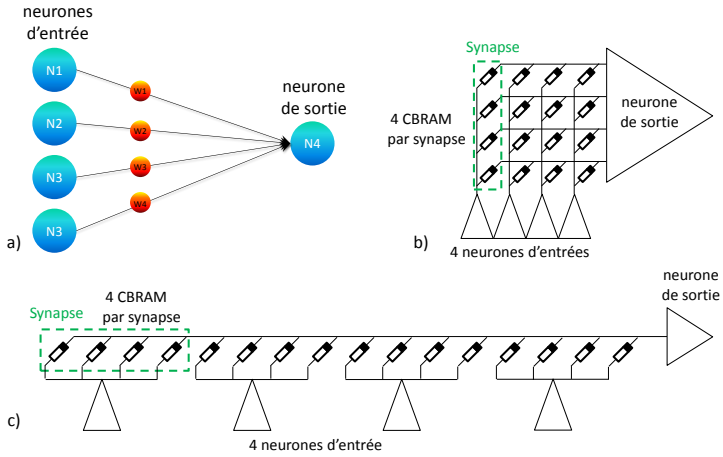


FIGURE 4.22 – a) Topologie du réseau simulé en VHDL-AMS – 4 neurones d'entrée, 1 neurone de sortie et quatre synapses. b) Implémentation avec des synapses organisées en suivant les anodes. c) Implémentation avec des synapses organisées en suivant les cathodes.

devrait être capable, grâce à la STDP, d'apprendre de façon non supervisée la corrélation de ces deux entrées. Après apprentissage, le neurone doit uniquement s'activer quand les entrées trois et quatre sont corrélées et les synapses correspondantes à ces entrées doivent être renforcées alors que les deux autres synapses seront affaiblies.

Ce réseau monocouche (figure 4.22 a)) comprend quatre neurones d'entrée, un neurone de sortie, ainsi que la méthode d'apprentissage STDP stochastique, implémentée par des générateurs de nombres pseudo-aléatoires pondérés (WPRNG). Cette simple application ne requiert pas une grande résolution synaptique, nous avons donc choisi une résolution de deux bits soit quatre CBRAM par synapse. Les CBRAM de chaque synapse peuvent être implémentées de deux manières différentes, soit en suivant les anodes (figure 4.22 b)), soit en suivant les cathodes (figure 4.22 c)).

Les méthodes de programmation stochastique sont présentées figure 4.23 pour les deux topologies envisageables (synapse organisée en suivant les anodes a) et b) et en suivant les cathodes c) et d)). Pour avoir une probabilité de commutation indépendante pour chaque CBRAM lors de la programmation, les synapses de la topologie en suivant les anodes doivent être programmées séquentiellement ce qui supprime le parallélisme naturel de ces réseaux. Dans notre simulation VHDL-AMS, nous avons donc implémenté la topologie suivant les cathodes qui permet de programmer en parallèle l'ensemble des synapses du neurone de sortie, mais qui nous oblige à retrouver une configuration crossbar (tous les transistors de sélections activés) en mode de lecture.

Le neurone d'entrée Avec cette topologie, le neurone d'entrée (figure 4.24 C)) est connecté à quatre anodes (autant que le nombre de CBRAM par synapse), il contient quatre multiplexeurs analogiques, un automate et deux WPRNG, l'un pondéré par la probabilité de LTP (probabilité qu'une cellule CBRAM effectue un SET quand sa synapse est dans la période LTP) et l'autre pondéré par la probabilité de LTD (probabilité qu'une cellule CBRAM effectue un RESET quand sa synapse est dans la période LTD). Les quatre multiplexeurs analogiques connectent chacune de ces anodes aux différentes tensions de lecture et de programmation ($Vread$, $Vset$, Vi, GND). Les multiplexeurs sont contrôlés par un automate. Les états de l'automate s'enchainent soit séquentiellement (d'une durée programmable par quatre registres), soit par l'activation de ce neurone d'entrée, soit par l'activation d'un neurone post-synaptique (figure 4.25). Cet automate a cinq registres :

– *State* Registre d'état – cinq états (trois bits) – IDLE, READ, LTP, SET, RESET

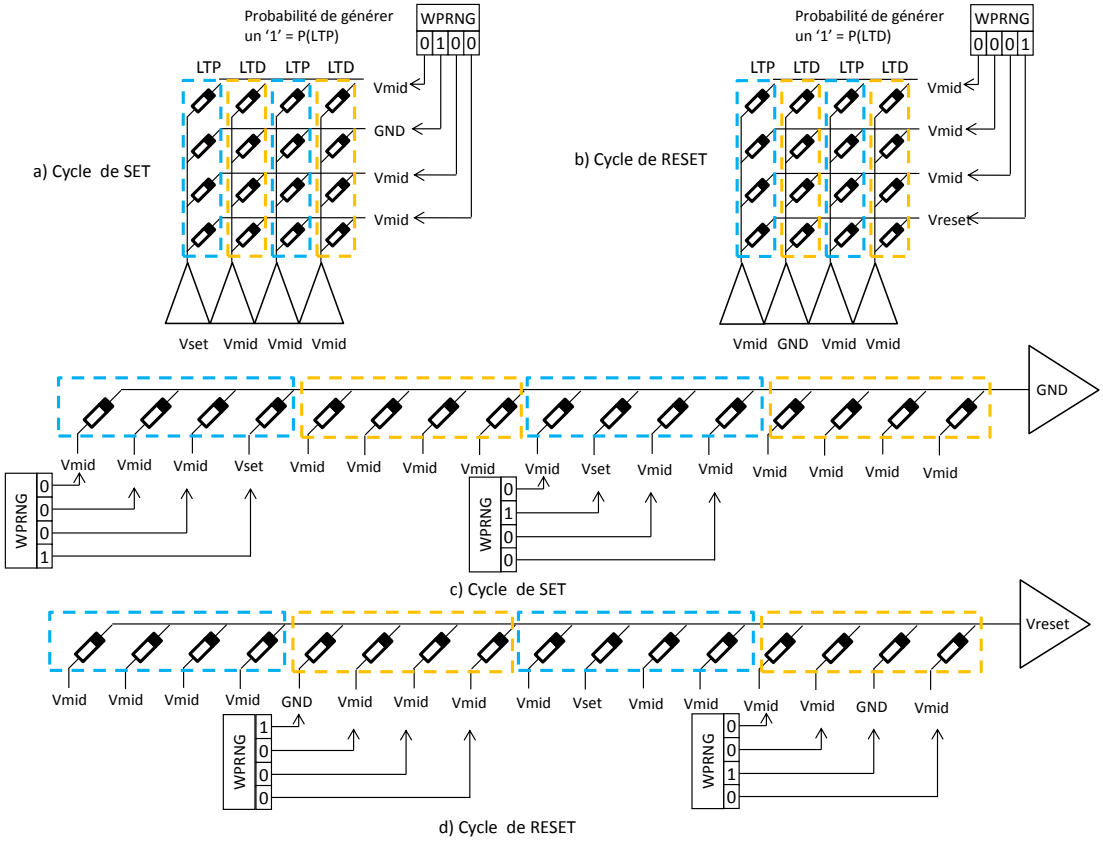


FIGURE 4.23 – Programmation stochastique sur les deux topologies synaptiques proposées figure 4.22. a) Cycle de SET et b) cycle de $RESET$ de la topologie suivant les anodes. c) Cycle de SET et d) cycle de $RESET$ de la topologie en suivant les cathodes. La topologie suivant les anodes ne permet pas de programmer les synapses en parallèle en ayant une probabilité de commutation indépendante pour chaque CBRAM ce qui est possible pour la topologie suivant les cathodes. Dans notre environnement de simulation, nous allons donc utiliser la topologie en suivant les cathodes.

- T_LTP Registre du compteur de temps LTP
- T_READ Registre du compteur de temps de l'impulsion de READ
- T_SET Registre du compteur de temps de l'impulsion de SET
- T_RESET Registre du compteur de temps de l'impulsion de RESET

En mode lecture, quand une entrée est activée, l'automate ouvre les multiplexeurs pour laisser circuler une impulsion de READ (0.1 V) d'une durée de $1\mu s$ puis cette entrée entre en période LTP. L'impulsion est alors pondérée par la valeur de la synapse, donc des quatre CBRAM la composant, et est intégrée par le neurone de sortie. Quand le neurone de sortie atteint son seuil d'activation, le neurone d'entrée entre alors en phase de programmation. L'automate envoi une commande de SET ou de tension intermédiaire (selon si cette synapse est dans la période LTP ou non) d'une durée $Tset$ suivie d'une commande de $RESET$ ou de tension intermédiaire (selon si cette synapse est dans la période LTD ou non) d'une durée $Treset$.

À la réception de la première commande (SET ou Vi), si la commande est SET (synapse en période LTP), chaque multiplexeur envoie une impulsion de $Vset$ ou Vi d'une durée $Tset$ en fonction de la sortie du WPRNG pondéré par la probabilité LTP (figure 4.23 c)). Si la commande correspond à la tension intermédiaire Vi (synapse en période LTD) alors Vi est envoyé d'une durée $Tset$.

À la réception de la seconde commande ($RESET$ ou Vi), si la commande est $RESET$ (synapse en période LTD), chaque multiplexeur envoie une impulsion de GND

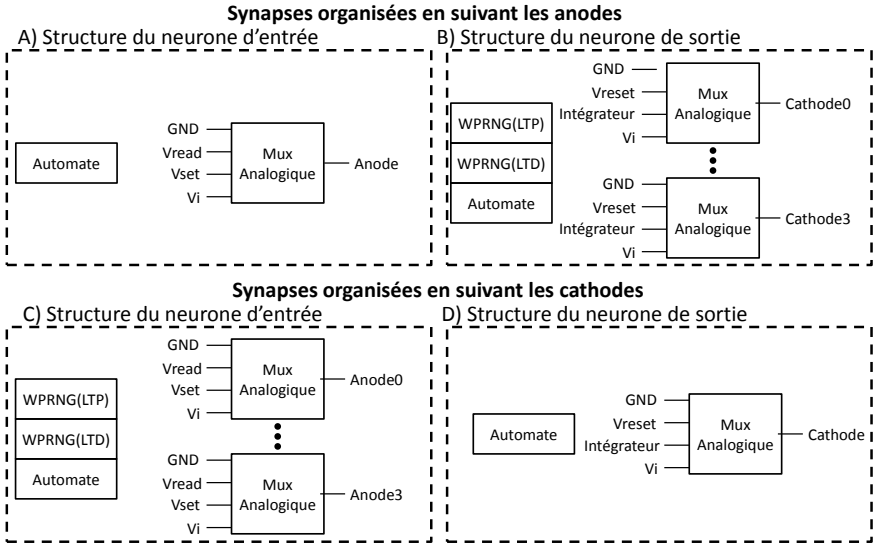


FIGURE 4.24 – A) Structure du neurone d'entrée et B) Structure du neurone de sortie de la topologie suivant les anodes. C) Structure du neurone d'entrée et D) Structure du neurone de sortie de la topologie suivant les cathodes.

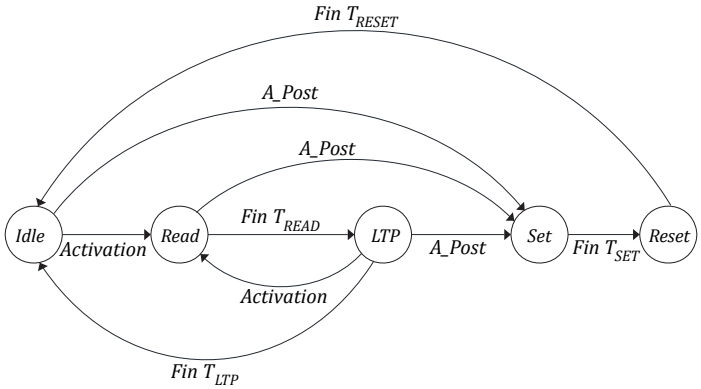


FIGURE 4.25 – Automate à cinq états du neurone d'entrée. *Activation* représente l'activation de ce neurone d'entrée et A_Post représente l'activation d'un neurone post-synaptique.

ou Vi d'une durée $Treset$ en fonction de la sortie du WPRNG pondéré par la probabilité LTD (figure 4.23 d)). Si la commande correspond à la tension intermédiaire Vi (synapse en période LTP) alors Vi est envoyé d'une durée $Treset$.

Le neurone de sortie Le neurone de sortie (figure 4.24 D)) contient un multiplexeur analogique, un automate, un intégrateur et un comparateur. Le multiplexeur analogique connecte la cathode aux différentes tensions de programmation (GND, Vreset, Vi) ainsi qu'à l'entrée de l'intégrateur en mode lecture. Les multiplexeurs sont contrôlés par un automate. Les états de l'automate s'enchainent soit séquentiellement (d'une durée programmable par quatre registres), soit par l'activation de ce neurone de sortie, soit par l'activation d'un autre neurone de sortie (figure 4.26). Cet automate à cinq registres :

- *State* Registre d'état – six états (trois bits) – READ, INHIBIT_I, INHIBIT, SET, RESET, REFRAC
- T_SET Registre du compteur de temps de l'impulsion de SET
- T_RESET Registre du compteur de temps de l'impulsion de RESET
- T_REFRAC Registre du compteur de temps de la période réfractaire

– T_INHIB Registre du compteur de temps de la période d'inhibition latérale

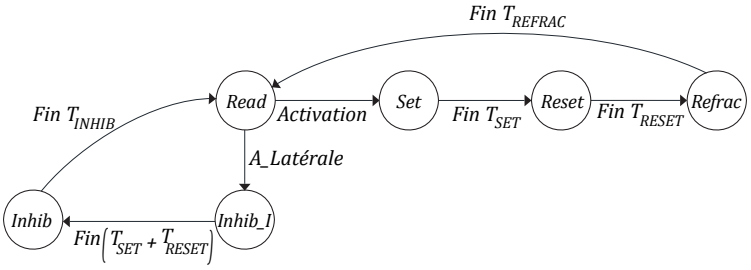


FIGURE 4.26 – Automate à 6 états du neurone de sortie. *Activation* représente l'activation de ce neurone de sortie et *A_latérale* représente l'activation d'un autre neurone de sortie présent sur la même couche.

En mode lecture, le neurone de sortie intègre le courant provenant de sa cathode, jusqu'à ce que son seuil d'activation Th soit atteint. Il entre alors en mode programmation (au même instant que les neurones d'entrée), et l'automate envoie, sur sa cathode, une impulsion de SET (GND) d'une durée T_{SET}, suivie d'une impulsion de RESET (V_{RESET}) d'une durée T_{RESET}.

Si un autre neurone de sortie présent sur la même couche se déclenche, alors notre neurone rentre en période d'inhibition. Dans un premier temps, il envoie une impulsion Vi sur sa cathode d'une durée $T_{SET} + T_{RESET}$ pour empêcher la programmation de ses synapses puis dans un second temps, termine sa période d'inhibition.

4.4.2 Simulation du réseau

Nous avons, tout d'abord, simulé notre réseau sur XNET ce qui nous a permis de trouver les paramètres du réseau avec un temps de simulation raisonnable. Ces paramètres, présentés table 4.2, permettent un apprentissage de la corrélation des entrées 3 et 4. La durée totale de la simulation est de $410ms$, nous avons fixé la période T_{LTP} à $2.5ms$ et la constante de fuite T_{LEAK} à $1ms$. Le seuil du neurone de sortie $I_{TH} = 111\mu A$ correspond à l'intégration du courant de lecture de 5 CBRAM à l'état ON (4,5 KΩ) avec $Vread = 0.1V$ pendant $1\mu s$ et avec un gain de 10^6 ($I_{ON} \times 5 \times 1\mu s \times 10^6 = 0,1/4500 \times 5 \times 10^{-6} \times 10^6 = 22.2\mu A \times 5 = 111\mu A$).

TABLE 4.2 – Paramètres neuronaux et synaptiques de la simulation VHDL-AMS.

Période LTP	T_{LTP}	2,5 ms
Seuil d'activation	I_{TH}	$111\mu A$
Constante de fuite	T_{LEAK}	1 ms
Période réfractaire	T_{REFRAC}	0.5 ms
Probabilité de LTP	P_{LTP}	35%
Probabilité de LTD	P_{LTD}	35%
Poids synaptique initial	W_{INIT}	0,8.WMAX
Nombre de CBRAM par synapse	Redun	4

Les résultats de la simulation VHDL-AMS sont présentés figure 4.27. On peut y retrouver l'activité des neurones d'entrée, l'intégration du neurone de sortie et son activité ainsi que les poids des synapses. Après apprentissage, les synapses des neurones d'entrées trois et quatre sont renforcées (poids maximaux de quatre, les quatre CBRAM

sont dans l'état LRS) alors que les synapses des neurones d'entrées un et deux sont affaiblies (poids minimaux de zéro, les quatre CBRAM sont dans l'état HRS). Le neurone de sortie est devenu sélectif à la corrélation des entrées trois et quatre.

4.5 Discussion et perspectives

Dans ce chapitre, nous avons étudié, au niveau circuit, la faisabilité d'une mémoire synaptique à base de dispositifs mémoires émergents et notamment, des dispositifs CBRAM. Nous avons montré que la structure crossbar génère de forts courants de fuite ($(N_{ON}-1)\frac{i_{set}}{2}$) qui, associés aux résistances parasites, limitent fortement la taille maximale de crossbar atteignable (875×875, $LRS = 100K\Omega$), et induisent une différence importante lors de la lecture des CBRAM (10%, $LRS = 4,5K\Omega$, 70×70). Les structures en matrices, quant à elles, permettent d'éliminer les courants de fuite et offrent au concepteur la possibilité d'implémenter un plus grand nombre de dispositifs (875×875, $LRS = 100K\Omega$). Entre les deux structures matricielles, la structure connectée par les cathodes est pour moi la structure la plus avantageuse, car elle permet de sélectionner les CBRAM d'un même neurone de sortie et, par conséquent, de programmer l'ensemble de ses synapses en parallèle sans se retrouver dans une configuration crossbar.

La solution en matrice est plus avantageuse en termes de consommation d'énergie et particulièrement avec des matrices de taille importante. D'autre part, les réseaux convolutionnels sont très efficaces, mais n'utilisent pas de grands champs synaptiques. Au contraire, ils utilisent un grand nombre de petits champs synaptiques qui eux pourraient être implémentés en crossbar, où la perte d'énergie serait compensée par la simplicité d'implémentation et le gain en superficie.

La prochaine étape, primordiale pour la création de puces neuromorphiques, est l'étude la co-intégration de ces mémoires émergentes dans procédé classique CMOS.

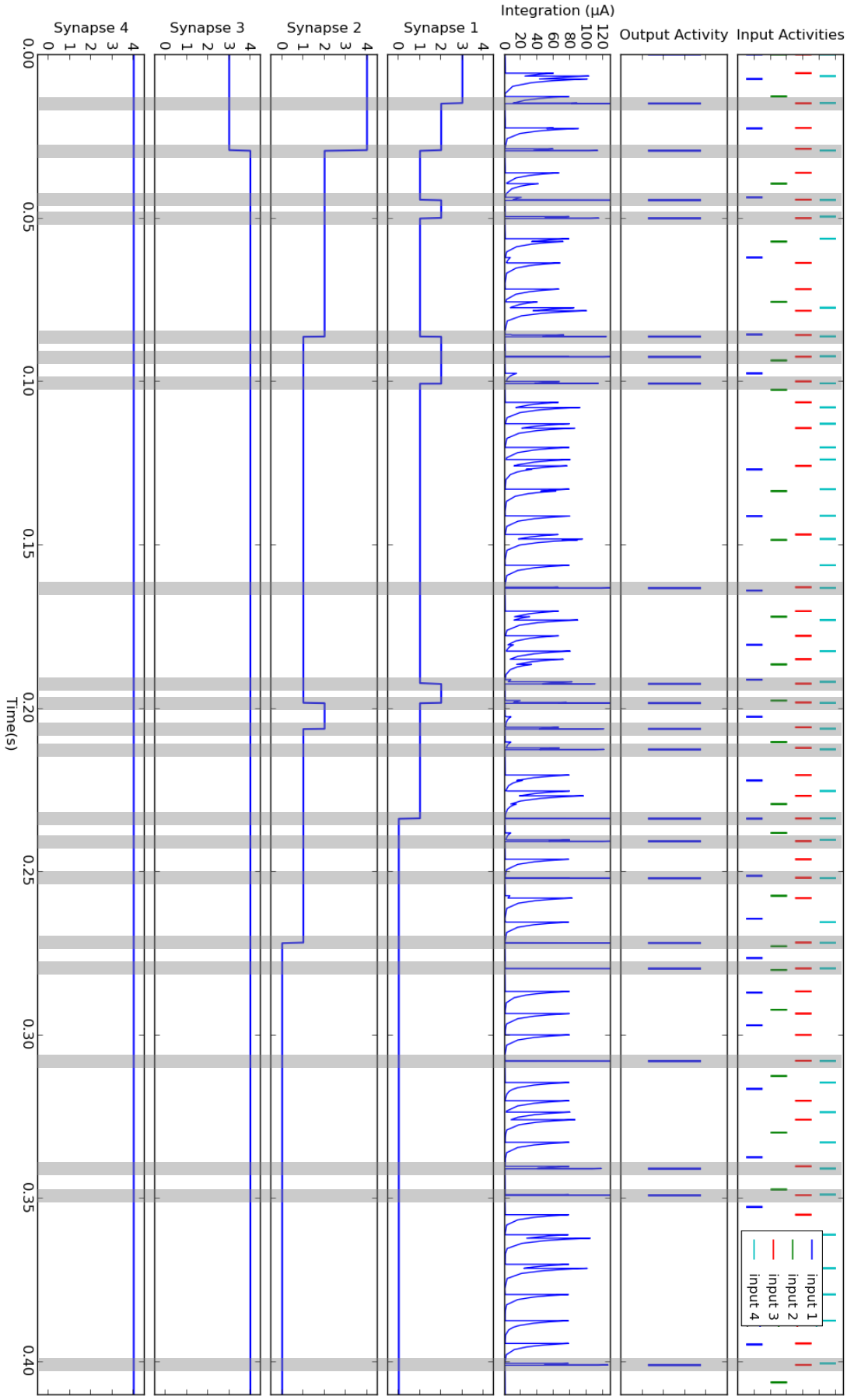


FIGURE 4.27 – Résultat de la simulation VHDL-AMS. Dans l'ordre : Activité des neurones d'entrée, Activité du neurone de sortie, Intégration du neurone de sortie, Poids des synapses une à quatre.

Chapitre 5

Co-intégration CMOS/CBRAM dédiée au Neuromorphique

Sommaire

5.1 Introduction 79
5.2 Choix des structures et des entrées/sorties 80
5.2.1 Modes de programmation et de lecture 81
5.2.2 Choix des entrées/sorties 82
5.3 Description de l'architecture 84
5.3.1 Architecture des circuits d'interface 84
5.3.2 Architecture haut niveau 85
5.3.3 Architecture du crossbar 86
5.3.4 Architecture de la matrice 86
5.3.5 Caractéristique du layout du circuit 87
5.4 Environnement de test 89
5.4.1 Circuit Imprimé 89
5.4.2 FPGA 90
5.5 Phases de tests 90
5.5.1 Test du circuit sous Eldo 91
5.5.2 Validation du circuit imprimé et du FPGA 91
5.5.3 Test du circuit 92
5.6 Discussion et perspectives 94

5.1 Introduction

La simple conception de mémoire synaptique ne peut suffire si celles-ci ne sont pas compatibles avec une technologie CMOS. En effet, il faut être capable de faire interagir les structures neuronales implémentées en CMOS avec les dispositifs memristifs pour pouvoir contrôler les impulsions de lecture ou de programmation circulant dans la mémoire synaptique. La réalisation d'un circuit co-intégrant ces technologies est donc une étape primordiale si l'on veut développer des réseaux neuromorphiques à base de technologies memristives. Pour valider l'implémentation d'une mémoire synaptique à base de technologies émergentes, nous avons réalisé puis testé une puce intégrant une technologie CMOS 130 nm et une technologie mémoire CBRAM. Ce circuit permet d'étudier les circuits d'interface entre les dispositifs memristifs et les circuits numériques. Il est donc contrôlé par des signaux numériques. Quatre topologies différentes de mémoire synaptique à base de CBRAM sont réalisées : une CBRAM seule, une CBRAM avec un

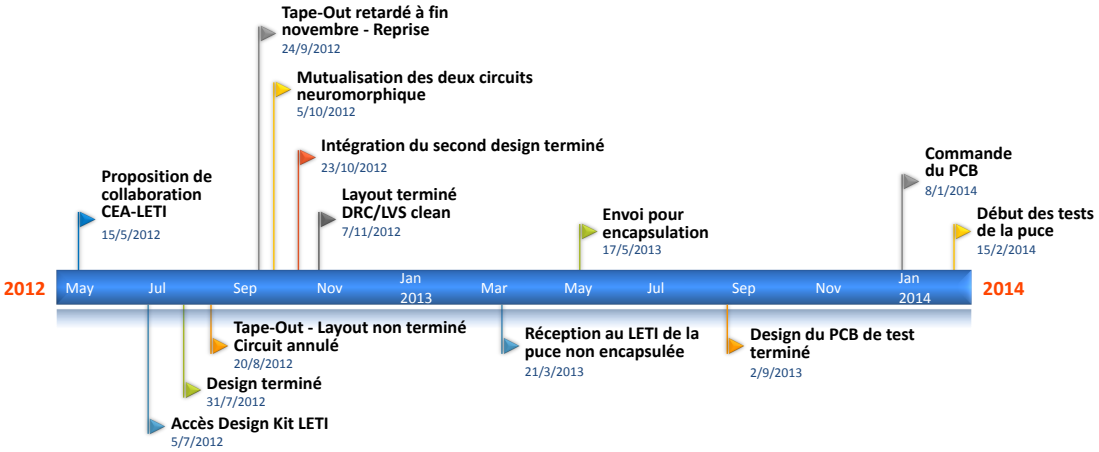


FIGURE 5.1 – Chronologie de la conception du circuit. Un an et neuf mois ont été nécessaires entre la proposition à la participation au MPW (Multi Project Wafer) et les tests du circuit. Durant cette période, un mois a été consacré à la conception du circuit et un mois à la réalisation du layout (réalisé par le CEA LETI).

transistor en série, un crossbar 4×4 et une matrice 4×4 connectée par les cathodes. Dans ce chapitre, nous présentons les différentes étapes de conception du circuit puis l'environnement et les résultats de test.

5.2 Choix des structures et des entrées/sorties

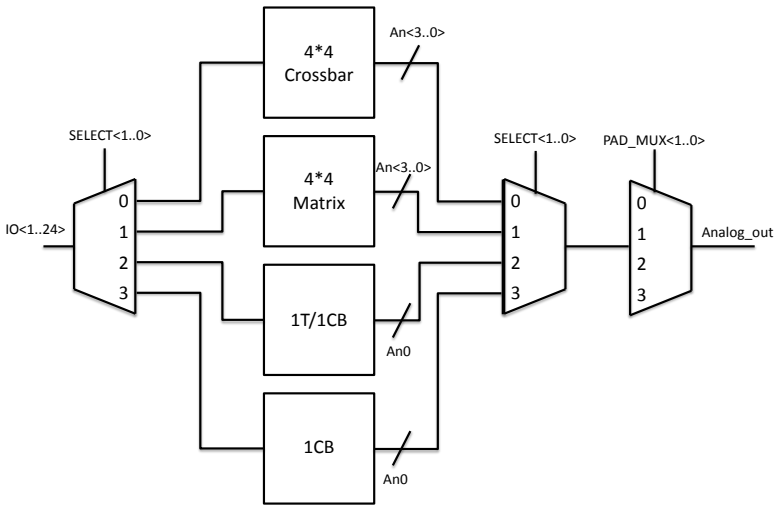


FIGURE 5.2 – Architecture haut niveau du circuit. En entrée, un multiplexeur analogique permet d'activer une des quatre topologies synaptiques (crossbar, matrice, 1T, 1T-1R) et deux multiplexeurs analogiques permettent de recueillir le courant de lecture d'une des dix cathodes.

Notre objectif initial était d'implémenter un réseau de neurones impulsionnels complet, comprenant une mémoire synaptique CBRAM et les circuits numériques associés aux neurones. Néanmoins, la date de lancement de la fabrication du circuit nous a contraint, dans un temps de conception limité, à nous concentrer sur les structures clés, les fonctions purement numériques pouvant être émulées a posteriori par un FPGA. En effet, un mois de développement nous a été accordé entre l'obtention du design kit et la livraison du schéma du circuit au CEA LETI pour la réalisation du layout (figure 5.1). Nous avons donc choisi de nous restreindre aux circuits d'interface CMOS sans y ajouter

les automates qui seront implémentés dans le FPGA qui contrôle le circuit. L'implémentation d'un crossbar 4×4 nous est suffisant pour réaliser l'application décrite dans le chapitre précédent. Afin de nous permettre l'étude de différentes topologies de mémoire synaptique, nous avons décidé d'ajouter une structure de matrice 4×4 connectée par les cathodes, et enfin deux cellules CBRAM uniques dont l'une possède un transistor en série. Les circuits d'interface CMOS sont implémentés par des multiplexeurs analogiques composés de portes de transmission, correspondant au modèle VHDL-AMS présenté dans le chapitre précédent, reliant une cathode ou une anode avec les entrées analogiques de la tension de lecture et des tensions de programmation. Le schéma haut niveau du circuit est présenté figure 5.2. On y retrouve les quatre structures synaptiques avec les multiplexeurs des entrées numériques. Pour spécifier correctement les besoins en termes de signaux d'entrées analogiques et numériques, nous allons étudier les différents cycles de programmation et de lecture des structures CBRAM incluses dans le circuit.

5.2.1 Modes de programmation et de lecture

Crossbar

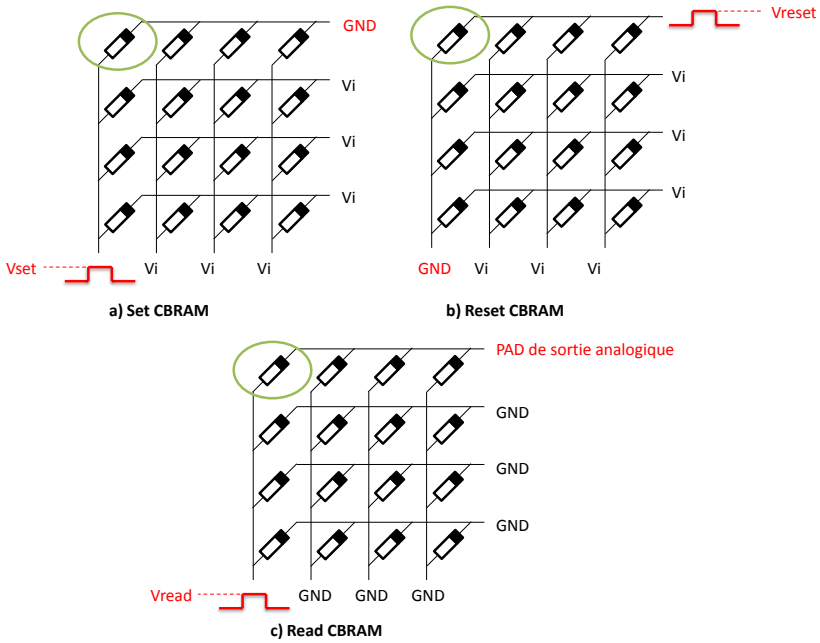


FIGURE 5.3 – Condition de programmation (a) RESET, b) SET) et de lecture c) d'un crossbar.

Les conditions de programmation et de lecture du crossbar sont présentées figure 5.3. Lors d'un SET, une impulsion d'amplitude V_{SET} est envoyée sur l'anode, et lors d'un RESET, une impulsion d'amplitude V_{RESET} est envoyée sur la cathode. Durant un SET ou un RESET, la tension intermédiaire Vi est envoyée sur chaque anode et cathode non sélectionnée. Lors de la lecture, une impulsion d'amplitude V_{READ} est envoyée sur une anode et une des cathodes est alors reliée au plot de sortie analogique pour recueillir le courant de lecture. Nous avons ajouté 4 modes de programmation pour le crossbar afin d'étudier les interactions d'impulsion (figure 5.4). Au total, il y a huit configurations (GND(repos), SET, RESET, READ (figure 5.3), LSET, HSET, LRESET, HRESET(figure 5.4)).

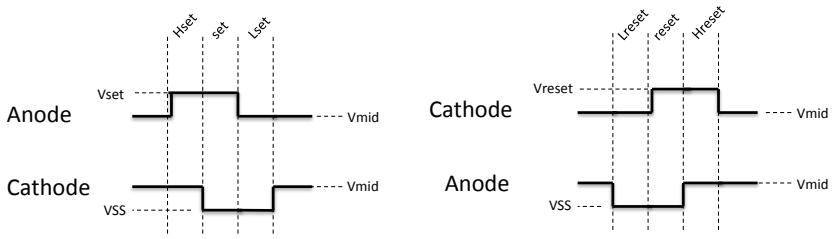


FIGURE 5.4 – Condition de programmation – interaction de pulses

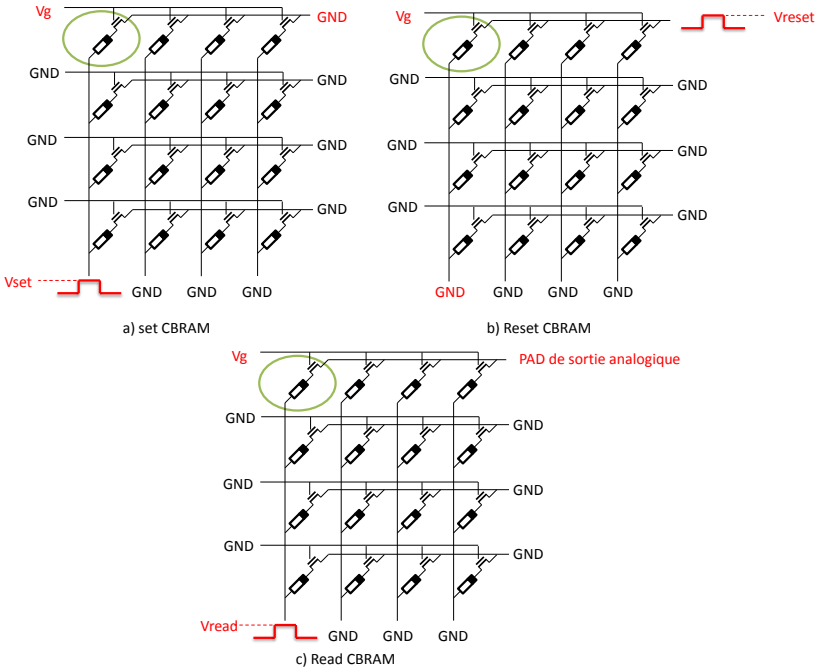


FIGURE 5.5 – Condition de programmation (a) RESET, b) SET) et de lecture c) de la matrice

Matrice connectée par les anodes

La matrice possède un transistor de sélection en série avec la cathode de chaque CBRAM. Leurs grilles sont connectées ensemble pour chaque ligne de la matrice. Les conditions de programmation et de lecture de la matrice sont présentées figure 5.5. On remarque que la tension intermédiaire Vi n'est plus nécessaire et que les interactions d'impulsion ne sont pas possibles. Pendant un SET, une impulsion d'amplitude V_{SET} est envoyée sur l'anode sélectionnée ainsi qu'une impulsion d'amplitude V_G sur la grille de la cathode sélectionnée. Pendant un RESET, une impulsion d'amplitude V_{RESET} est envoyée sur la cathode sélectionnée ainsi qu'une impulsion d'amplitude V_G sur la grille de cette même cathode. Pendant, un READ, une impulsion d'amplitude V_{READ} est envoyée sur l'anode sélectionnée ainsi qu'une impulsion d'amplitude V_G sur la grille de la cathode sélectionnée et le courant résultant est redirigé vers le plot de sortie analogique.

5.2.2 Choix des entrées/sorties

Nous disposions de 23 plots d'entrée-sortie dont huit ont été consacrés aux entrées analogiques (table 5.1).

Nous avons souhaité pouvoir sélectionner les anodes ou les cathodes séparément ou simultanément pour étudier la commutation de multiples dispositifs en parallèle. Nous avons dédié huit entrées numériques pour sélectionner les quatre anodes et les quatre

TABLE 5.1 – Plots analogique – Sept entrées et une sortie.

VDD	Tension d'alimentation CMOS
GND	Masse
$Vset$	Tension de SET
$Vres$	Tension de RESET
Vi	Tension intermédiaire pour le crossbar
$Vgate$	Tension de grille des transistors de sélection
$Vread$	Tension de READ
$Analog\_out$	Sortie analogique du courant de lecture

cathodes ($An < 3..0 >$ et $Ca < 3..0 >$ table 5.2).

TABLE 5.2 – Plots de sélections des anodes et cathodes.

$An < 3..0 >$	Sélection des anodes
$Ca < 3..0 >$	Sélection des Cathodes
$SELECT < 1..0 >$	Choix de la structure à étudier
$MODE < 2..0 >$	Choix mode de programmation (Détaillé dans la suite du chapitre)
$PAD\_MUX < 1..0 >$	Contrôle du multiplexeur analogique reliant une cathode à la sortie analogique du courant de lecture

Sachant que ce circuit possède quatre structures de CBRAM différentes, nous avons dédié deux entrées numériques pour la sélection de la structure à étudier ($SELECT < 1..0 >$ table 5.2).

Afin de contrôler les multiplexeurs analogiques et de pouvoir choisir, soit un des modes de programmation, soit le mode de lecture, trois entrées numériques ont été utilisées ($MODE < 2..0 >$ table 5.2).

Trois entrées du circuit restent disponibles pour capturer le courant de lecture. J'ai choisi d'utiliser deux de ces entrées pour contrôler un multiplexeur analogique 4 :1 qui connecte une des quatre cathodes à l'entrée restante ($SELECT < 1..0 >$ table 5.2).

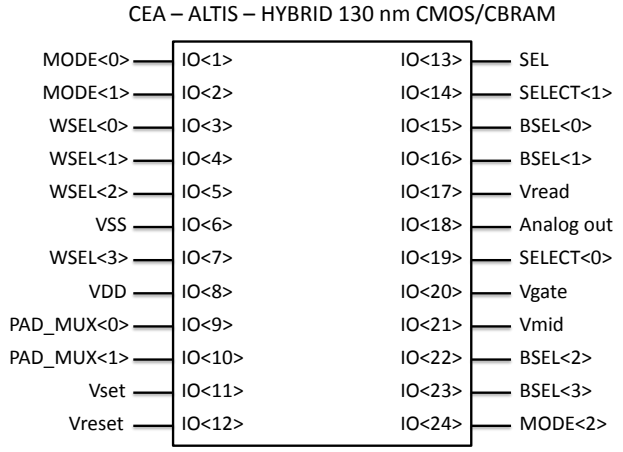


FIGURE 5.6 – Brochage du circuit neuromorphique co-intégrant la technologie CMOS 130 nm et la technologie mémoire émergente CBRAM.

5.3 Description de l'architecture

5.3.1 Architecture des circuits d'interface

Le circuit d'interface est l'unité de base de notre circuit. Il sert à transmettre une impulsion électrique vers une anode ou vers une cathode. Ce circuit d'interface doit être contrôlable par un signal numérique provenant de la logique CMOS de chaque anode ou cathode. Nous avons choisi d'implémenter le circuit d'interface avec une porte de transmission (figure 5.7). Cette porte de transmission est connectée d'un coté à une des entrées analogiques ($Vread$ ou $Vset$ ou $Vreset$ ou Vi ou GND) et de l'autre, à l'une des anodes ou cathodes. Pour contrôler une porte de transmission efficacement, les signaux logiques la contrôlant doivent être compris entre la masse (GND) pour la logique '0' et la tension la plus élevée pour la logique '1'. Or, les tensions de programmation peuvent être supérieures à la tension d'alimentation VDD. Nous avons alors décidé d'ajouter un montage élévateur de tension (translateur de niveaux) entre la logique et la porte de transmission (figure 5.8). La taille de la porte de transmission doit être correctement définie en fonction des courants de programmation. Dans notre circuit, nous avons choisi de sur-dimensionner les portes de transmission pour garantir la commutation de l'ensemble des dispositifs CBRAM en parallèle. Grace au modèle de simulation fourni par le CEA LETI, nous avons évalué le courant de programmation d'un dispositif à 400 μA. Pour pouvoir programmer l'ensemble des dispositifs d'une même anode ou cathode, une porte de transmission doit être capable de fournir 4 $\times$ 400 mA ou 1,6 mA. Nous avons surdimensionné chaque porte de transmission pour fournir un courant de 7 mA, ce qui nous permet d'avoir une marge d'erreur au cas où la résistance LRS des dispositifs CBRAM soient inférieurs à celle attendu (LRS $\approx$1 KΩ).

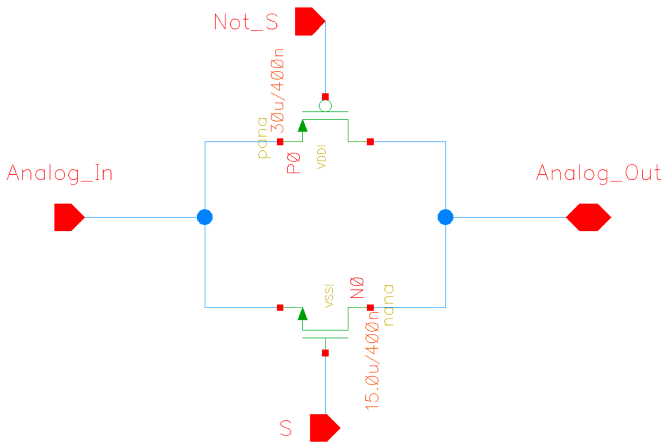


FIGURE 5.7 – Schéma de la porte de transmission implémentant un circuit d'interface.

Comme énoncé dans le paragraphe précédent, les circuits d'interface ont été surdimensionnés afin d'assurer un apport en courant suffisant à la commutation de l'ensemble des dispositifs. Les portes de transmission sont composées de deux transistors analogiques, un NMOS et un PMOS. Ces transistors analogiques supportent des tensions plus élevées que les transistors standards (3,3V au lieu de 1,5V). Le transistor analogique NMOS a un courant de saturation de 513 $\mu A/\mu m$ et le transistor analogique PMOS a un courant de saturation de 238 $\mu A/\mu m$. Bien qu'avec nos calculs, 1,6 mA était suffisant pour commuter l'ensemble des dispositifs, nous avons choisi de pouvoir fournir plus de 7 mA. Nous avons implémenté un transistor NMOS de 15 μm (15 $\times$ 513 = 7,7 mA) ainsi qu'un transistor PMOS de 30 μm (30 $\times$ 238 = 7,15 mA). Toutes les portes de transmission incluses dans notre circuit possèdent ce dimensionnement pour un gain en temps de conception. Le layout d'une porte de transmission et d'un

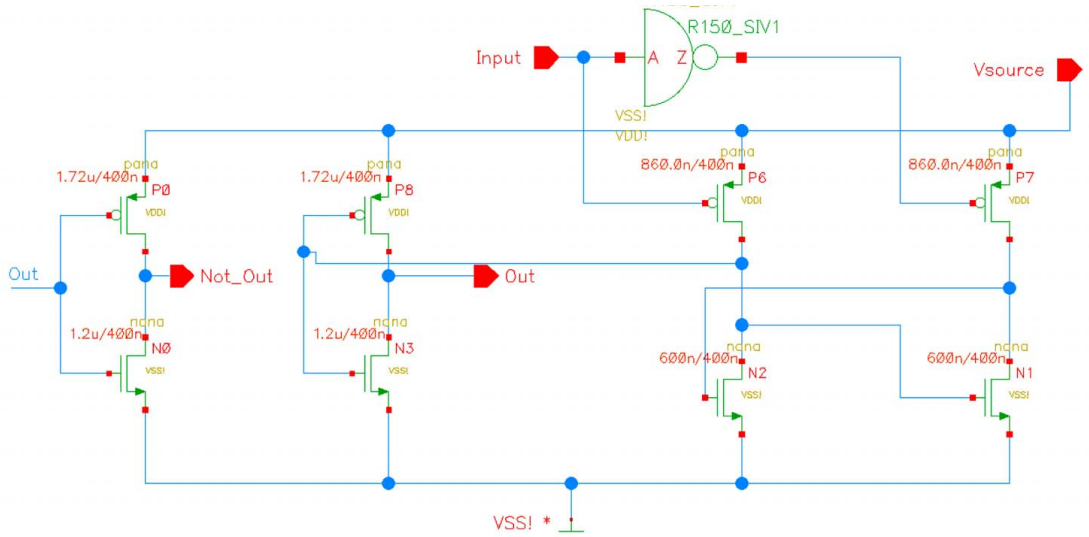


FIGURE 5.8 – Schéma du circuit translateur de niveau logique.

montage élévateur de tension est présenté figure 5.9. Nous avons fait ce choix de surdimensionnement car les conditions de programmation et notamment l'étape de forming (qui peut requérir des tensions supérieures) n'étaient pas complètement caractérisées et pouvaient changer selon la technologie CBRAM incluse dans la puce.

Pour nos prochains circuits, nous pourrons nous servir des résultats de nos tests sur le circuit pour optimiser l'ensemble des portes de transmission en fonction des différents courants de programmation (i_{SET}, i_{RESET}) et de lecture (i_{READ}) pour un gain en superficie. Par exemple, la porte de transmission qui laisse circuler l'impulsion de READ doit être de taille inferieure car les courants de lecture sont bien inférieurs aux courants de programmations.

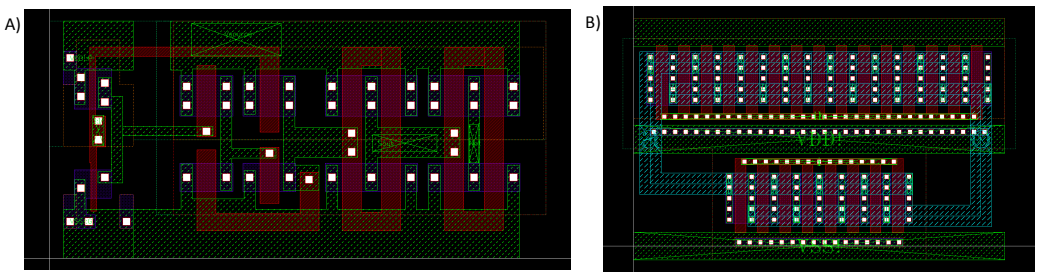


FIGURE 5.9 – Layout a) d'un montage élévateur de tension et b) d'une porte de transmission.

5.3.2 Architecture haut niveau

La figure 5.2 présente le schéma haut niveau du circuit. On peut y retrouver les quatre topologies synaptiques et trois multiplexeurs. Un multiplexeur analogique, contrôlé par le signal $SELECT < 1..0 >$, permet d'activer l'une des quatre topologies synaptiques (table 5.3).

Les deux multiplexeurs de sortie permettent de choisir l'une des cathodes lors de la lecture. Le premier, contrôlé par $SELECT < 1..0 >$, sélectionne les cathodes d'une topologie et le second, contrôlé par $PAD\_MUX < 1..0 >$, sélectionne l'une des quatre cathodes (table 5.4).

TABLE 5.3 – Table de vérité du signal $SELECT < 1..0 >$: sélection des topologies.

$SELECT < 1 >$	$SELECT < 0 >$	topologie
0	0	1T
0	1	1T-1R
1	0	Matrice
1	1	Crossbar

TABLE 5.4 – Table de vérité du signal $PAD\_MUX < 1..0 >$: sélection de la cathode à lire.

$PAD\_MUX < 1 >$	$PAD\_MUX < 0 >$	Cathode
0	0	Cathode 0
0	1	Cathode 1
1	0	Cathode 2
1	1	Cathode 3

5.3.3 Architecture du crossbar

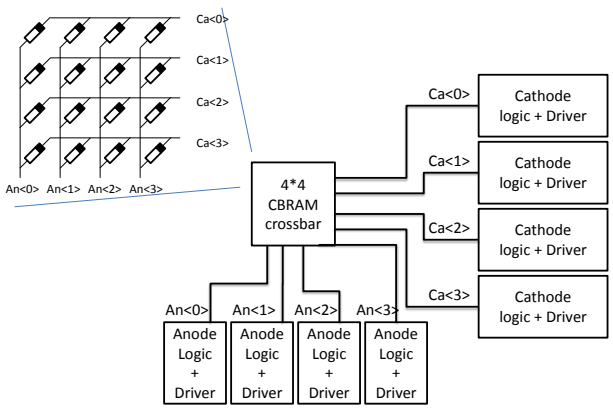


FIGURE 5.10 – Architecture du circuit crossbar. Ce circuit contient le crossbar de CBRAM 4×4 ainsi que les circuits d'interface et la logique contrôlant chaque anode et cathode (figure 5.11).

Le circuit crossbar (figure 5.10) contient le crossbar 4×4 de CBRAM ainsi que les circuits d'interface et le circuit logique contrôlant chaque anode et cathode. Comme énoncé dans la section précédente, les circuits d'interface sont implémentés par des portes de transmission qui sont contrôlées (ouvertes ou fermées) par le circuit logique associé à une anode ou une cathode. Pour chaque anode (figure 5.11 a), il y a quatre portes de transmission ayant les tensions $Vset$, $Vread$, GND et Vi. Pour chaque cathode (figure 5.11 b), il y a quatre portes de transmission ayant les tensions $Vreset$, GND et Vi ainsi que l'accès au plot de sortie pour la lecture. Ces portes sont contrôlées par le circuit logique en fonction des signaux d'entrées En ('1' si la topologie crossbar est sélectionnée), SEL ('1' si cette anode est sélectionnée) et $MODE < 2..0 >$. Ce dernier signal permet de choisir les impulsions qui vont circuler dans le crossbar en fonction du mode désiré. Il existe huit différents modes dans le crossbar : GND(repos)-READ-SET-RESET-LRESET-HRESET-LSET-HSET. La table de vérité de la logique contrôlant les portes de transmission de chaque anode et de chaque cathode est présentée en annexe table 1.

5.3.4 Architecture de la matrice

Le circuit de la matrice (figure 5.10) contient une matrice 4×4 de CBRAM dont les grilles sont connectées par les cathodes ainsi que les circuits d'interface et les circuits

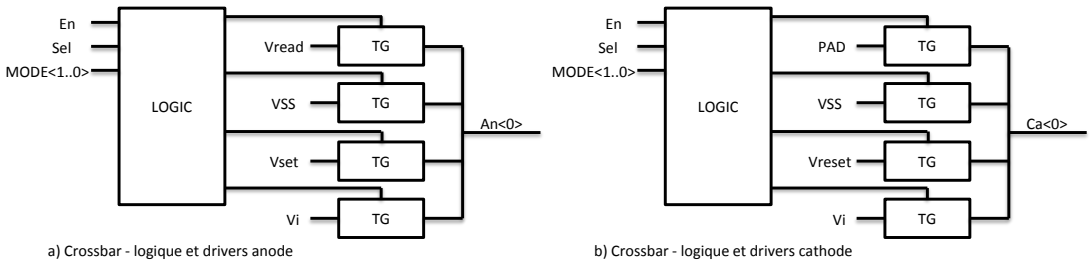


FIGURE 5.11 – Architecture des circuits d'interface du crossbar. a) logique et circuits d'interface d'une anode du crossbar, quatre portes de transmission permettent de faire circuler une impulsion sur cette anode avec des tensions différentes ($Vset$, $Vread$, GND, Vi). b) logique et circuits d'interface d'une cathode du crossbar, quatre portes de transmission permettent de faire circuler une impulsion sur cette anode avec des tensions différentes ($Vreset$, GND, Vi) ou de connecter cette cathode au plot de sortie de la puce (An_out) lors de la lecture. La logique permet d'ouvrir une porte de transmission à la fois en fonction des signaux En, SEL et $MODE < 2..0 >$ (table 1)

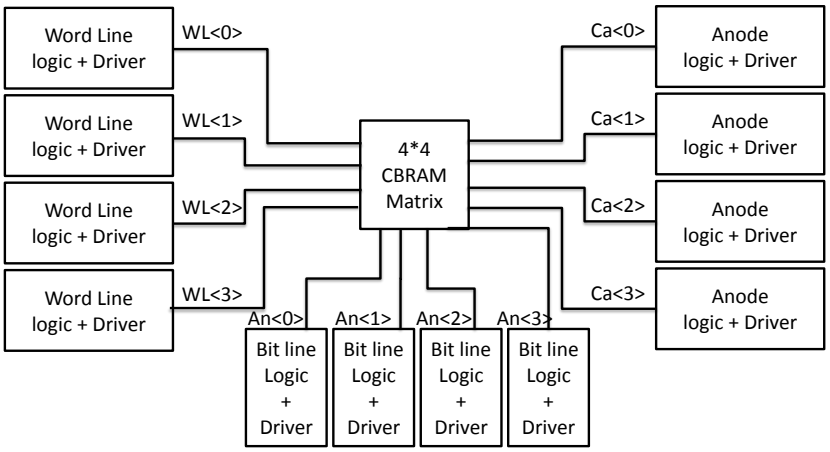


FIGURE 5.12 – Architecture du circuit de la matrice. Ce circuit contient la matrice de CBRAM 4×4 ainsi que les circuits d'interface et la logique contrôlant chaque anode et cathode (figure 5.13).

logiques contrôlant chaque anode, cathode et ligne de grille. Pour chaque anode (figure 5.13 a), il y a quatre portes de transmission ayant les tensions $Vset$, $Vread$, GND et $Vreset$. Pour chaque cathode (figure 5.13 b), il y a trois portes de transmission ayant les tensions $Vreset$ et GND ainsi que l'accès au plot de sortie pour la lecture. Pour chaque ligne de grille (figure 5.13 c), il y a une porte de transmission ayant la tension Vg. Ces portes sont contrôlées par le circuit logique en fonction des signaux d'entrées En ('1' si la topologie matrice est sélectionnée), SEL ('1' si cette anode-cathode-grille est sélectionnée) et $MODE < 2..0 >$. Ce dernier signal permet de choisir les impulsions qui vont circuler dans le crossbar en fonction du mode désiré. Il existe quatre différents modes dans le crossbar : GND(repos)-READ-SET-RESET. La table de vérité de la logique contrôlant les portes de transmission de chaque anode, chaque cathode et de chaque ligne de grille est présentée en annexe table 2.

5.3.5 Caractéristique du layout du circuit

Nous avons utilisé un procédé CMOS 130 nm, intégralement intégré au-dessous de la couche CBRAM. La longueur minimale d'un transistor standard est de 120 nm et celle d'un transistor analogique est de 400 nm. Les circuits de la porte de transmission et du montage élévateur de tension ainsi que les transistors de sélection de la matrice

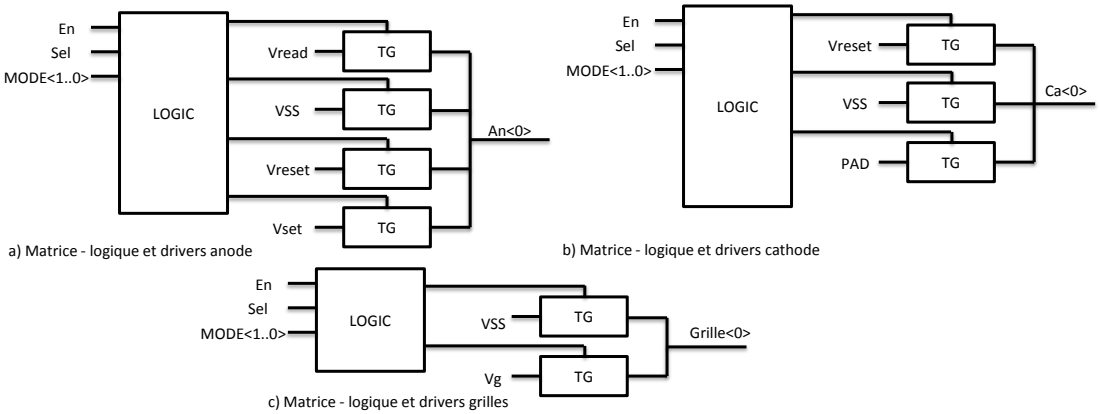


FIGURE 5.13 – Architecture des circuits d'interface de la matrice. a) Logique et circuits d'interface d'une anode de la matrice, quatre portes de transmission permettent de faire circuler une impulsion sur cette anode avec des tensions différentes ($Vset$, $Vread$, VSS, $Vreset$). b) Logique et circuits d'interface d'une cathode de la matrice, quatre portes de transmission permettent de faire circuler une impulsion sur cette anode avec des tensions différentes ($Vreset$, VSS) ou de connecter cette cathode à la sortie An_out lors de la lecture. c) Logique et circuits d'interface d'une ligne de grille de la matrice, deux portes de transmission permettent de faire circuler une impulsion sur cette anode avec des tensions différentes (Vg ou VSS). La logique permet d'ouvrir une porte de transmission à la fois en fonction des signaux En, SEL et $MODE < 2..0 >$ (table 2).

sont exclusivement constitués de transistors analogiques. Tous les autres circuits sont constitués de transistors standards. Le layout du crossbar et de la matrice de CBRAM a été développé dans la section 4.3.4 du chapitre précédent.

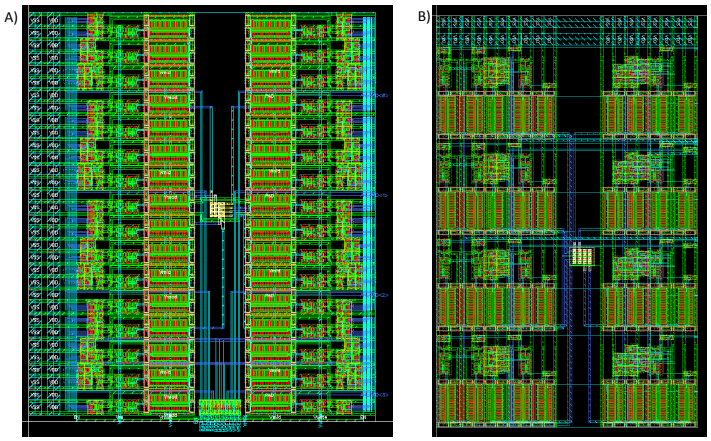


FIGURE 5.14 – a) layout du circuit du crossbar correspondant au schéma de la figure 5.10 b) layout du circuit de la matrice correspondant au schéma de la figure 5.12.

Le layout du circuit du crossbar (figure 5.14 a) correspond au schéma de la figure 5.10 et le layout du circuit de la matrice (figure 5.14 b) correspond au schéma de la figure 5.12. Les dimensions du circuit du crossbar sont 131,5 $\mu m \times 111,45 \mu m$ = 1 505 μm^2 et les dimensions du circuit de la matrice sont 141 $\mu m \times 88,8 \mu m$ = 12 520 μm^2. On remarque immédiatement la taille imposante du CMOS comparée au crossbar (ou à la matrice) de dispositifs CBRAM. Cela peut, en partie, s'expliquer par le surdimensionnement des nombreuses portes de transmissions (32 pour le circuit du crossbar et 36 pour le circuit de la matrice) ainsi que par la taille du procédé CMOS utilisé. Il est légitime de s'interroger sur la faisabilité de l'implémentation de large crossbar ou matrice. En effet, même si le procédé CMOS n'est pas à l'état de l'art, le

procédé CBRAM ne l'est pas non plus (200 nm). Des travaux ont montré des CBRAM de 30 nm (Sakamoto et al. (2004)) et 20 nm (Kund et al. (2005)) et un autre démontre que les CBRAM pourraient atteindre la taille d'un filament d'atomes, soit inférieur à 5 nm (Jameson et al. (2012)). Le problème de la taille des circuits d'interface et de la logique CMOS comparée à la taille du crossbar va donc s'étendre aux procédés CMOS à l'état de l'art. Le second problème est lié à la forte connectivité. Bien que le rapport des surfaces CMOS/CBRAM soit déjà important, il ne faut pas oublier que les neurones ne sont pas inclus dans notre circuit. Les neurones augmenteraient la taille du CMOS comparée à la taille des dispositifs memristifs.

5.4 Environnement de test

5.4.1 Circuit Imprimé

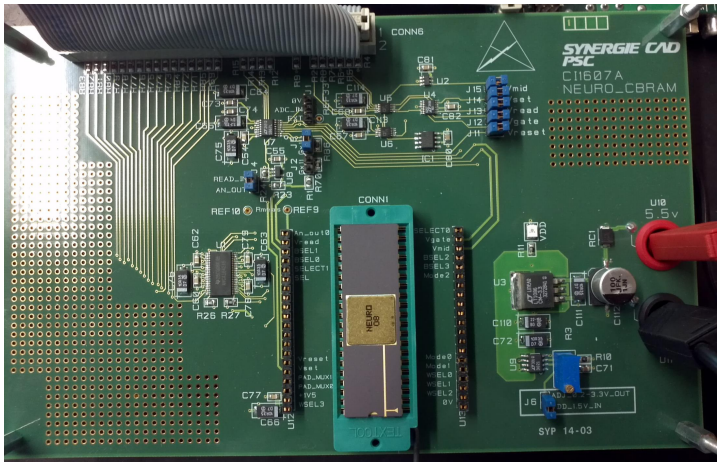


FIGURE 5.15 – Circuit imprimé de test avec le circuit neuromorphique au centre.

Pour le test et la validation de notre circuit neuromorphique, nous avons développé un circuit imprimé spécialisé pouvant accueillir la puce et contrôler celle-ci grâce à un FPGA sur une carte de prototypage. Le circuit imprimé a pour rôle de fournir les différentes tensions de programmation, de lecture et d'alimentation du circuit (VDD, GND, $Vread$, $Vset$, $Vreset$, Vi, Vg) mais aussi d'adapter les signaux logiques provenant du FPGA vers la puce. En effet, la carte de prototypage fournit des signaux numériques 3,3 V et le circuit ne peut recevoir que des signaux compatibles avec sa tension d'alimentation VCC, compris entre 1,5 et 3,0 V.

Le schéma du circuit imprimé est présenté en annexe 1. Le circuit imprimé est alimenté en 5V, nous avons intégré un régulateur 3,3 V pour alimenter les composants du circuit imprimé, suivi d'un régulateur variable de 1,4 V à 3,3V pour alimenter le circuit (VDD) et régler les niveaux logiques d'entrée (notamment pour les tensions élevées nécessaires au forming).

Nous avons utilisé un translateur de niveaux pour convertir les signaux logiques 3,3 V provenant du FPGA de la carte de prototypage vers le niveau VDD (1,4V à 3,3V). Les sorties du convertisseur sont directement connectées aux entrées de la puce.

Pour fournir les tensions de programmation et de lecture, nous utilisons six convertisseur numérique analogique (CNA 16 bits) (0V - 3,3V) suivi de six buffers pour assurer un niveau de courant suffisant. Les CNA sont contrôlés par le FPGA avec une liaison série SPI.

Les derniers éléments du circuit imprimé servent à mesurer le courant lors de la lecture d'une ou de plusieurs CBRAM. Nous avons connecté la sortie analogique de la

puce à un amplificateur opérationnel suivi d'un convertisseur analogique numériques (CAN) 16 bits. La valeur convertie sur 16 bits est alors envoyée vers le FPGA à travers une liaison série SPI.

5.4.2 FPGA

Un ordinateur de contrôle envoie des commandes grâce à un script Python à travers une UART vers la carte de prototypage (ML402) contenant le FPGA (Virtex-4). Dans le FPGA, nous avons implémenté un processeur Microblaze ainsi qu'un module spécialisé VHDL dont le rôle est de contrôler les signaux numériques à envoyer au circuit (figure annexe 2). Le Microblaze reçoit les commandes de l'UART qui peuvent être de deux types : mise à jour des tensions et choix du mode de programmation ou de lecture.

Mise à jour des tensions de programmation et de lecture Il est possible de modifier les différentes tensions de programmation et de lecture en envoyant une commande de SET_VOLTAGE vers le Microblaze. Cette commande possède 2 arguments, l'adresse du CNA (quelle tension est à mettre à jour ?) ainsi que l'amplitude de la tension. Ces informations sont ensuite envoyées par le SPI vers le CNA et la mise à jour est instantanée.

Mode de programmation et de lecture du circuit Pour effectuer une opération de programmation ou de lecture sur le circuit, il faut envoyer trois commandes au Microblaze au travers de l'UART avec un script python qui seront transmises au module VHDL contenant une simple machine à états dont le code est retranscrit en annexe 1. Cette machine à états permet de contrôler les signaux arrivant au circuit. Une première commande CMD sur 15 bits contient toutes les informations concernant les entrées numériques du circuit. Une seconde contient la variable « var_tmp_low » sur 32 bits et la dernière contient la variable « var_tmp_up » sur 32 bits. Ces variables permettent d'envoyer le signal MODE séparément des autres signaux afin de limiter les pics de courants dans le circuit lors de l'ouverture et de la fermeture des portes de transmissions. La machine à états possède 5 états :

1. attente une nouvelle commande
2. envoi de tous les signaux (An, Ca, PAD_MUX, SELECT) vers le circuit sauf le signal « MODE » pendant le temps « var_tmp_low ».
3. envoi du signal « MODE » pendant le temps « var_tmp_up ». En mode lecture, le signal « conv_now » est envoyé au Microblaze qui envoie une commande SPI à l'CAN pour convertir le courant de lecture.
4. remise à zéro du signal « MODE » pendant le temps « var_tmp_low ».
5. remise à zéro de tous les autres signaux

La valeur du courant de lecture est reçue par le Microblaze via le SPI et est envoyée via l'UART vers le script Python où le calcul de la résistance équivalente de la CBRAM lue est réalisé.

5.5 Phases de tests

Nous pouvons découper les phases de tests en trois parties. Tout d'abord, nous avons testé le circuit grâce au simulateur Eldo sous Cadence. Puis, nous avons validé le fonctionnement du circuit imprimé et enfin, nous avons réalisé les tests du circuit physique.

5.5.1 Test du circuit sous Eldo

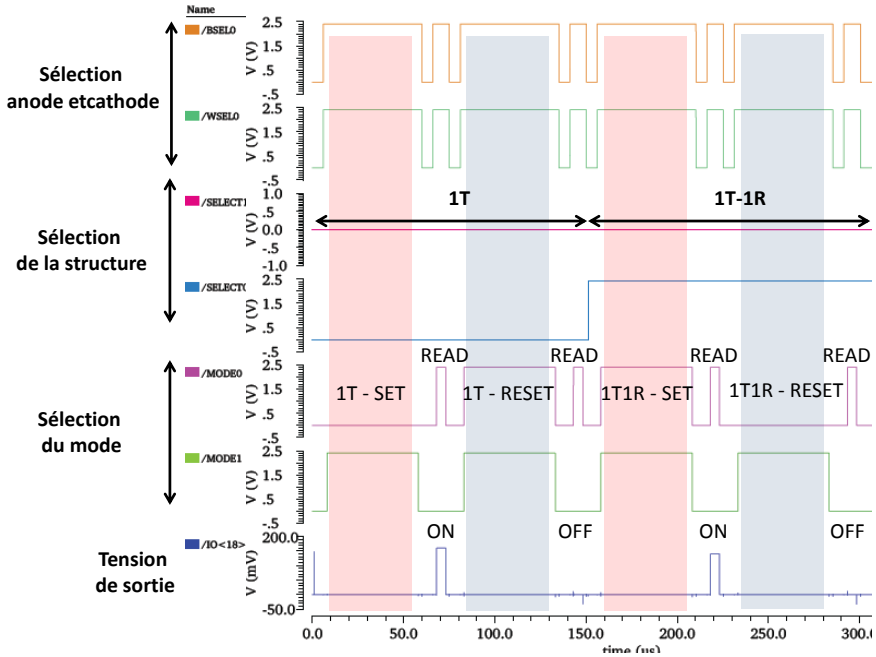


FIGURE 5.16 – Simulation Eldo des circuits 1R et 1T1R post-extraction, nous effectuons les cycles de SET-READ-RESET-READ sur chaque structure et observons la commutation des deux dispositifs CBRAM .

Les simulations sous Eldo ont permis de vérifier le bon fonctionnement du circuit lors de chaque étape du développement. La figure 5.16 présente les résultats d'une simulation du circuit post-extraction réalisée sur les deux structures les plus simples, 1R et 1T1R. Dans cette simulation, nous sélectionnons tout d'abord la structure 1R et nous effectuons les cycles de SET-READ-RESET-READ, puis nous répétons ces cycles sur la structure 1T-1R. On peut voir sur la broche de sortie IO<18> du circuit que le dispositif est de faible résistance après un SET et de forte résistance après un RESET. Nous avons placé une résistance de $1K\Omega$, reliée à la masse, en série avec la sortie afin de visualiser une tension proportionnelle au courant de lecture se qui nous permet de déterminer la valeur de résistance du dispositif R selon la formule :

$$R = \frac{(V_{READ} - V_{<IO>18}) * 1000}{V_{<IO>18}} \tag{5.1}$$

Nous utilisons dans cette simulation une tension V_{READ} de 1V et nous obtenons une tension de sortie de 157,75 mV pour la structure 1R, soit une résistance de 5 340 Ω, et 136,42 mV pour la structure 1T1R, soit une résistance de 6 330 Ω.

La figure 5.17 présente une seconde simulation post-extraction intéressante, la programmation parallèle de l'ensemble des dispositifs du crossbar puis la lecture séquentielle de chaque dispositif CBRAM. On retrouve les résultat de différence de courants de lecture due aux courants de fuite présenté dans le chapitre 4 section 4.3.1.

5.5.2 Validation du circuit imprimé et du FPGA

La validation du circuit imprimé s'est effectuée en plusieurs étapes. Nous avons tout d'abord vérifié les tensions d'alimentation des deux CNA, du CAN et du translateur de niveaux logique (3,3V) ainsi que le régulateur de tension qui alimente le circuit (VDD) et ses signaux d'entrée. Ensuite, nous avons vérifié et validé la programmation des deux

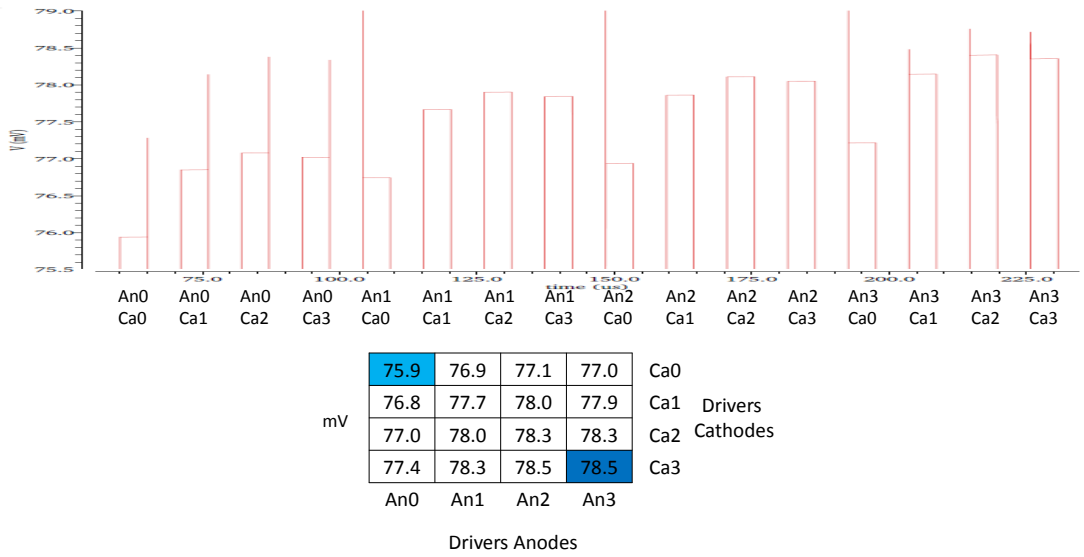


FIGURE 5.17 – Eldo : Simulation post-extraction du crossbar, nous avons effectué un set sur l'ensemble des dispositifs du crossbar puis une lecture séquentielle de chacun des dispositifs. On retrouve les résultats présentés dans le chapitre 4 section 4.3.1 de variation de courants de lecture due aux courants de fuite dans le crossbar.

CNA en sélectionnant des niveaux de tensions différentes pour chaque entrée analogique ($Vread$, $Vset$, $Vreset$, Vi, Vg). Nous avons validé le CAN en connectant directement une sortie du CNA au CAN, et nous avons vérifié que le script python recevait bien la valeur de la tension de sortie du CNA. Nous avons placé une résistance entre la sortie du CNA et l'entrée du CAN et vérifié que nous pouvons retrouver la valeur de celle-ci dans le script python. Enfin, nous avons validé le translateur de niveau logique (3,3V du FPGA vers la tension VDD du circuit).

Pour la lecture, nous avons utilisé des impulsions comprises entre 0.1V et 1V. Nous envoyons 10 impulsions de lecture successives mais espacées de plusieurs dizaines de millisecondes et convertissons les 10 valeurs de lecture grâce au CNA. La valeur reçue par le script Python est la moyenne de ces 10 conversions.

Afin de vérifier la précision de nos impulsions nous avons envoyé deux commandes de programmation et enregistré les résultats à l'aide d'un oscilloscope. La figure 5.18 montre que nous pouvons contrôler efficacement la durée d'une impulsion durant un SET et un RESET. Dans ces deux exemples, nous envoyons des impulsions de durée différentes (5 μ s, 1 μ s et 500 ns) et nous pouvons voir que ces durées sont respectées aux bornes du circuit.

5.5.3 Test du circuit

Ces étapes réalisées nous avons pu commencer les tests des circuits, vingt circuits ont été livrés au CEA LETI dont douze ont été mis en boîtier puis livrés dans nos locaux. Les huit circuits restant sont destinés à être testés à l'aide d'une station de test sous pointes mais le temps imparti ne nous l'a pas permis.

Lors de la réception des circuits, nous avons appris que le fondeur a malheureusement changé de technologie CBRAM entre le tape-out et la fabrication du circuit, invalidant en partie le modèle utilisé durant sa conception. Sur les six circuits (dont le notre) incluent dans le MPW (Multi Project Wafer) seul notre circuit a été testé pour le moment ce qui ne nous permets pas de pouvoir comparer nos résultats de test.

Ce nouvel empilement CBRAM requiert une étape de forming, ce qui n'était pas le cas de la technologie précédente et a nécessité d'adapter le protocole et le PCB de

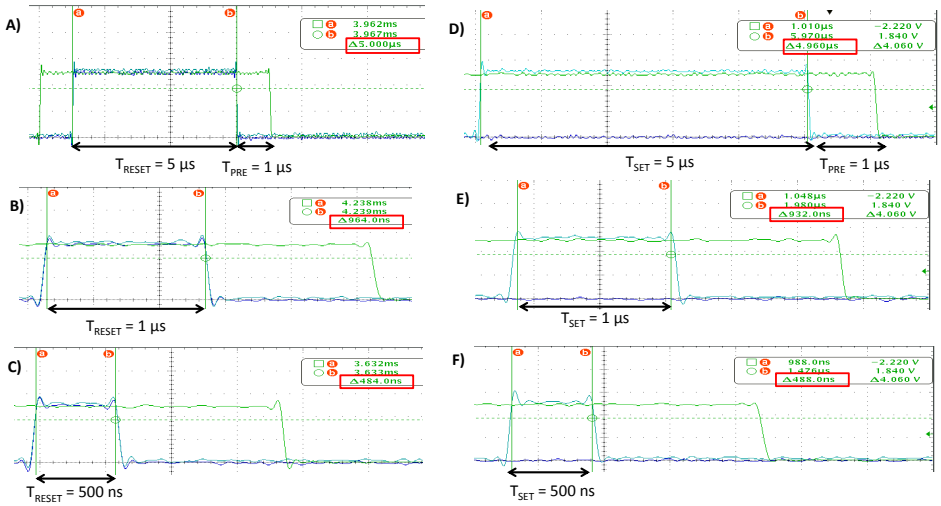


FIGURE 5.18 – Exemples de commande SET et RESET à l'oscilloscope sur la structure 1R avec des durées d'impulsions, T_{SET} différentes (A) 5 μs, B) 1 μs, C) 500 ns) et T_{RESET} différentes (D) 5 μs, E) 1 μs et F) 500 ns). Le signal vert est la sélection de l'anode An<0> qui intervient 5 μs avant et après le signal MODE, le signal bleu clair est le signal MODE<0> qui est à '1' en mode SET/RESET et le signal bleu foncé est le signal MODE<1> qui est à '0' en mode SET et à '1' en mode RESET. On peut voir que le module FPGA permet de contrôler efficacement chaque signal d'entrée du circuit.

test. Durant cette étape, une tension de programmation plus élevée est appliquée sur les électrodes du dispositif afin d'initialiser la formation du pont conducteur. On nous a conseillé d'appliquer des impulsions d'amplitude augmentant graduellement (de 2.0 V à 3.0 V) de 1 μs jusqu'à ce que la résistance de la cellule CBRAM atteigne une valeur inferieure à 10 KΩ.

La figure 5.19 montre une mesure suggérant l'observation d'une étape de forming réalisé sur l'un des dispositifs CBRAM du crossbar ($An<0>, Ca<0>$). On peut voir la résistance interne du dispositif diminuer (de >2,5 MΩ à 500 KΩ) au fur et à mesure de la création du filament conducteur. Pour ce forming, nous avons préféré utiliser des tensions de SET de 2,0 V mais sur de longues périodes de 10 ms. Nous avons pu observer ce comportement sur plusieurs dispositifs atteignant des valeurs de résistance inferieurs à 10 KΩ. Malheureusement, lors de nos essais nous avons pu observer deux difficultés récurrentes. Dans un premier temps, beaucoup de dispositifs ne se formaient pas, c'est sa dire que leur résistance interne n'évoluait pas et restait supérieure à la dizaine de$M\Omega$. Dans un second temps, nous avons observé ce qui semblait être l'étape de forming sur certain dispositifs mais il était impossible d'effectuer un RESET sur ces dispositifs qui restaient dans un état LRS de l'ordre du $K\Omega$.

Nous avons tout de même réussi à réaliser des commutations sur certains dispositifs mais avec des niveaux de résistance qui ne sont pas ceux attendus. En effet, ces CBRAM devraient avoir une LRS inferieure à 10 KΩ et une HRS supérieure à 1 MΩ. La figure 5.20 montre des cycles de SET-READ-RESET-READ sur différents dispositifs. On peut voir qu'une commutation s'effectue entre les cycles de SET et RESET mais avec un ΔR très petit (entre 1 KΩ et 200KΩ). Les conditions de programmations de ces exemples sont présentées tableau 5.5. On remarque que les durées d'impulsion sont supérieures à celles attendue, de l'ordre de la milliseconde au lieu d'être de l'ordre de la microseconde.

La difficulté majeure fut de retrouver des conditions de programmations permettant une commutation des dispositifs CBRAM. Les résultats de test de notre circuit montrent des commutations que l'on pourrait qualifier de partielles. Néanmoins, ces mesures sont reproductibles et nous avons écarté toutes possibilités d'effets capacitifs

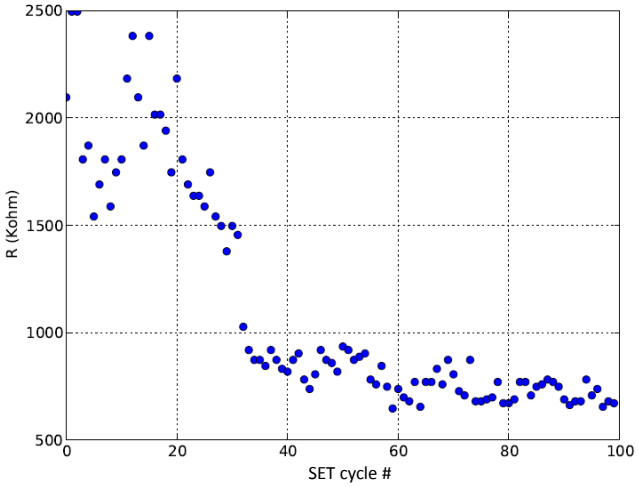


FIGURE 5.19 – Cycle de forming d'une CBRAM du crossbar. La résistance interne du dispositif diminue (>2,5 MΩ à 500 KΩ au fur et à mesure de la création du filament conducteur. Pour ce forming, nous avons préféré utiliser des tensions de SET identique de 2,0 V mais avec des durées d'impulsions plus importante (10 ms).

en espaçant les impulsions de lecture de plusieurs dizaines de millisecondes.

TABLE 5.5 – Condition de programmation des résultats présenté figure 5.20.

	Vset	Tset	Vreset	Treset	Vread	VG	Structure	An	Ca
A)	2,2	10 ms	2,2	10 ms	0,5	X	Crossbar	0	0
B)	2,3	10 ms	2,3	10 ms	0,5	X	Crossbar	0	0
C)	1,8	1 ms	1,8	1 ms	1,0	2,2	Matrix	2	0
D)	1,8	10 ms	1,8	10 ms	1,0	2,0	Matrix	2	0
E)	2,2	5 ms	2,2	5 ms	1,0	X	Crossbar	1	2
F)	2,2	5 ms	2,2	5 ms	1,0	X	Crossbar	1	2

5.6 Discussion et perspectives

Dans ce chapitre, nous avons présenté le développement et les phases de test du premier circuit neuromorphique co-intégrant une technologie CMOS 130 nm avec une technologie mémoire émergente, la CBRAM. Nous présentons une méthode à base de porte de transmission pour contrôler les impulsions circulant dans les structures de dispositifs CBRAM. Après la vérification de l'environnement de test, nous avons observé des commutations reproductibles aussi bien dans le crossbar que dans la matrice.

Ce premier travail fut une étape primordiale dans le développement de circuit neuromorphique. Dans un premier temps, il nous a permis d'étudier les différentes règles de dessins intervenant lors de l'intégration de dispositifs CBRAM avec un procédé CMOS classique. Notamment, nous avons vu que les règles du procédé CMOS utilisé ne nous permettaient pas de réaliser une intégration maximale de la CBRAM. En effet, bien qu'une cellule CBRAM seule possède une surface de 1,63 μm^2, son intégration avec notre procédé CMOS multiplie par 40 cette surface quand la CBRAM est intégrée dans un crossbar ou par 58 quand la CBRAM est intégrée dans une matrice (résultats présen-

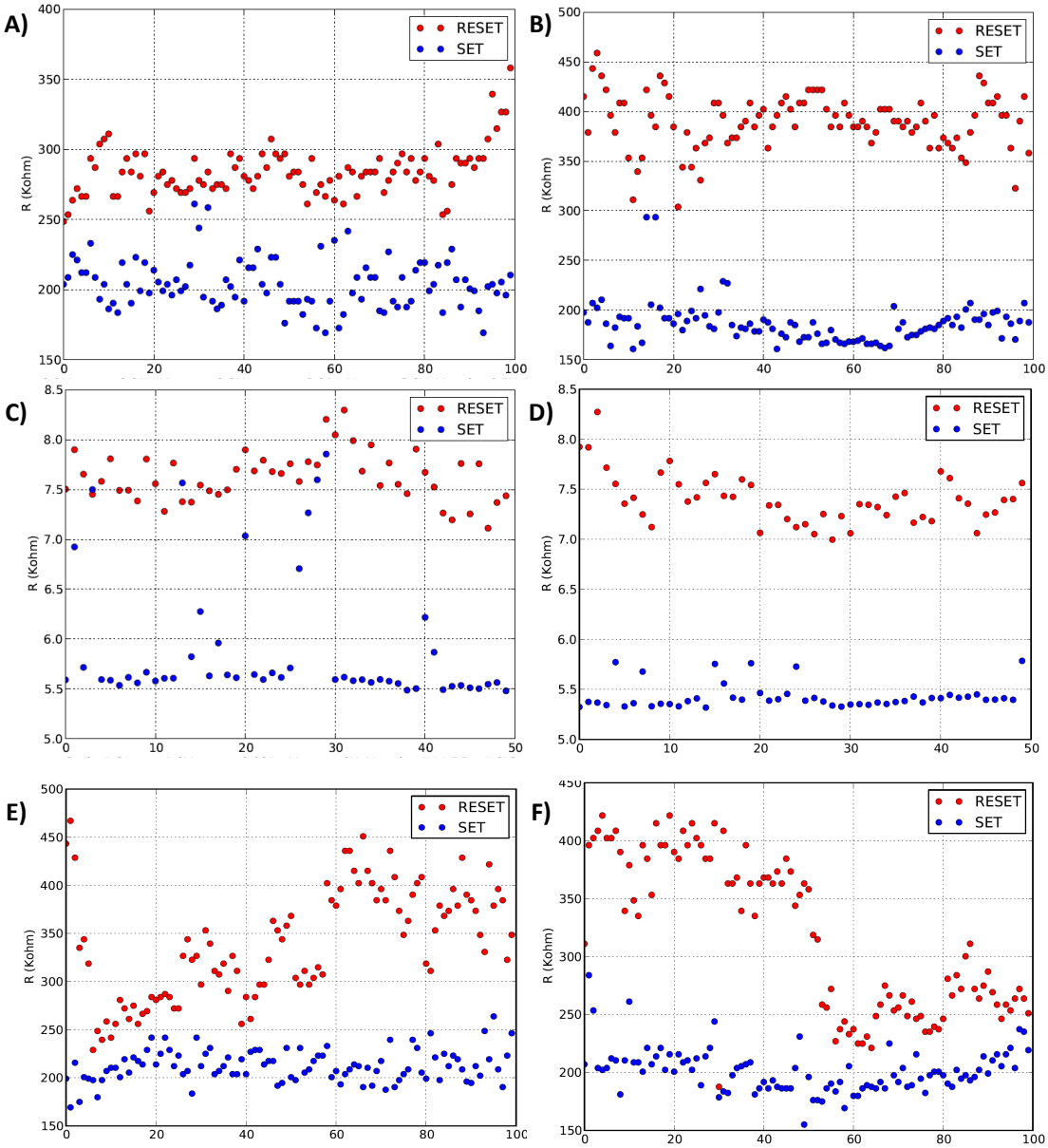


FIGURE 5.20 – Démonstration de commutations partielles sur certain dispositifs du crossbar et de la matrice. Les conditions de programmation sont présenté tableau 5.5.

tés section 4.3.4). L'utilisation d'un procédé CMOS à un nœud technologique minimale est donc requis s'il l'on souhaite atteindre une densité d'intégration maximale.

Dans un second temps, ce circuit nous a permis de réaliser une estimation de la surface CMOS nécessaire aux contrôles des structures de dispositifs CBRAM en crossbar et en matrice. Bien que chaque porte de transmission soit sur-dimensionnée, il faut aussi prendre en compte que les circuits des neurones ne sont pas présents. Pour une architecture neuronal numérique, parallèle et complète, il faudrait par exemple ajouter un circuit de conversion analogique vers numérique pour la lecture des CBRAM, un accumulateur pour le potentiel membranaire ainsi qu'une machine à états analogue à celle implémentée dans notre FPGA, et ce pour chaque neurone. Si l'on considère une CBRAM par synapse, cela signifie qu'il existe un neurone pour chaque anode et chaque cathode. Il est donc imaginable que l'espace CMOS alloué à chaque neurone serait alors d'une taille surdimensionnée comparée à la structure synaptique et qu'un problème de connectivité se poserait (notamment une limitation dans la longueur des lignes d'accès).

La commutation contrôlée et reproductible de plusieurs dispositifs memristifs dans les structures en crossbar et en matrice marque pour nous la réussite de ce projet. La perspective suivante est maintenant l'implémentation des neurones avec une structure synaptique à base de dispositifs memristifs. C'est d'ailleurs l'un des objectifs principaux de la thèse du prochain doctorant de notre laboratoire ce qui démontre la volonté de réaliser un système neuromorphique complet à base de dispositifs mémoires émergents.

Chapitre 6

Conclusion et perspectives

Sommaire

6.1 Conclusion 97
6.2 Perspectives 98

6.1 Conclusion

Dans cette thèse, nous avons tout d'abord cherché à optimiser un réseau de neurones impulsionnels afin de permettre son intégration dans une architecture purement numérique.

L'optimisation de la résolution du poids synaptique permet de diminuer la superficie et l'énergie consommée par la mémoire implémentant les synapses. Pour aller plus loin, dans le cas où la mémoire synaptique serait implémentée par de la mémoire classique de type SRAM, DRAM, on peut imaginer un système de mémoire configurable pour atteindre une résolution synaptique minimale selon l'application visée. Ce serait alors un atout sur le plan de la consommation d'énergie en imaginant une mise en veille prolongée de la partie de la mémoire synaptique non utilisée, en utilisant par exemple des techniques de basse consommation comme le power gating ou le clock gating.

Ensuite, si l'on considère une architecture entièrement parallèle où chaque neurone calcule et met à jour son potentiel membranaire individuellement, nous ne pouvons alors pas considérer un quelconque multiplexage des ressources de calcul. Dans ce cas, chaque neurone doit posséder un ALU implémentant un calcul exponentiel pour la fuite de son potentiel membranaire ce qui n'est pas envisageable dans une architecture où la superficie est une contrainte majeure. Notre étude, qui démontre que le remplacement du calcul exponentiel par un calcul linéaire n'affecte ni l'apprentissage ni la fonctionnalité du réseau neuronal, permet donc de réaliser une seconde optimisation essentielle dans notre objectif d'architecture embarqué.

Enfin, le stage de Vincent Lorrain démontre des difficultés rencontrées lors de l'implémentation d'une STDP temporelle dans une architecture numérique. Les résultats positifs de ce démonstrateur montrent que les optimisations proposées permettent cette implémentation tout en conservant une efficacité aussi bien du point de vue computationnelle que de l'apprentissage, ce qui est progrès important par rapport aux précédentes implémentations de STDP temporelle qui sont peu efficaces (ex : dans Spinnaker)

Les résultats de cette première phase d'optimisation permettent d'envisager l'intégration d'un réseau de neurones impulsionnel dans un système compatible avec le monde de l'embarqué.

La seconde phase de notre projet est l'implémentation de la mémoire synaptique par des dispositifs mémoires émergents de type memristifs.

Nous avons commencé par comparer différentes structures synaptiques à base de dispositifs CBRAM. Bien que la structure en crossbar présente une densité d'intégration maximale, nous avons quantifié par simulation que cette structure est énergivore, est limitée en taille du fait des chutes de tension, possède une forte variation du courant de lecture et enfin, engendre des perturbations sur les cellules voisines lors des phases de programmation. L'introduction de transistors de sélection permet de résoudre ces différents points à l'exception de la variation du courant qui peut être réduit mais non pas éliminé. Nous en sommes venus à la conclusion que la structure en matrice connectée par les cathodes est la plus intéressante, car elle permet notamment une programmation parallèle des dispositifs. Nous avons aussi noté que la forte densité d'intégration du crossbar pourrait compenser la consommation d'énergie élevée dans le cas d'une mémoire synaptique de taille raisonnable.

Enfin, la proposition du LETI de participer au design d'un circuit neuromorphique à base de dispositifs memristifs CBRAM a été une grande valeur ajoutée pour cette thèse. En effet, cela nous a permis d'acquérir une première expérience enrichissante de design avec ce type de dispositifs memristifs. Mais surtout, les résultats positifs des tests de notre circuit ont permis de valider la co-intégration d'une technologie CMOS avec des dispositifs memristifs.

Ce travail de thèse apporte une réponse favorable au développement de réseaux impulsionnels dédiés au monde de l'embarqué en vue de leur intégration dans un imageur 1) en démontrant qu'il était possible de simplifier ces réseaux tout en conservant leur efficacité computationnelle, 2) en quantifiant la densité d'intégration, la consommation d'énergie et le parallélisme qu'offrent les dispositifs memristifs dans différentes topologies d'intégration et enfin 3) en validant la faisabilité de l'implémentation de mémoire synaptique à base de dispositifs memristifs intégrés avec un procédé CMOS classique.

6.2 Perspectives

La suite logique de mes travaux amène à la conception d'un circuit neuromorphique complet comprenant des neurones et des synapses à base de dispositifs memristifs. Je conseillerais au concepteur d'implémenter un système numérique qui aurait l'avantage d'être plus facilement paramétrable qu'un système analogique. Pour la conception des neurones, je prendrais pour exemple les modèles présentés dans le chapitre 4. Pour le circuit de lecture, j'utiliserais un circuit comparateur (miroir de courant) pour déterminer l'état ON ou OFF (ou multi-niveaux) du dispositif memristif et additionner la valeur binaire du dispositif à un accumulateur CMOS. Ce circuit serait alors une première mondiale et une grande valorisation pour notre laboratoire.

Je voudrais maintenant développer trois directions de travaux de recherche, relatif à l'architecture des réseaux impulsionnels, qui me tiennent à cœur.

Dans un premier temps, le fait d'implémenter ce type de réseau biologique sur des architectures synchrones est pour moi une mauvaise direction prise par la majorité de la communauté. En effet, les réseaux de neurones biologiques sont naturellement asynchrones ce qui permet aux neurones d'évoluer indépendamment les uns des autres. Dans une architecture synchrone, l'asynchronisme naturel des réseaux biologique est émulé en augmentant la fréquence de l'horloge globale. Cela a pour effet d'augmenter considérablement la consommation d'énergie du système, alors que les réseaux biologiques n'évoluent qu'à une faible fréquence de l'ordre du Hertz ou kilohertz. Je proposerai donc d'étudier l'implémentation des réseaux impulsionnels avec une architecture asynchrone dédiée aux systèmes basse puissance comme la logique NCL (Null Convention Logique)

ou MTNCL (Multi-Threshold NCL) (Smith and Di (2009); Roclin (2010)). Ce type d'architecture serait, à mon avis, intéressant pour implémenter des réseaux évoluant à si faible fréquence. Cette technologie possède un grand potentiel, car elle a l'avantage d'être robuste, d'être faible consommation (les transitions d'états n'ont lieu que lors de traitement utile), d'être insensible au retard intrinsèque au circuit (pas de contrainte de timing, de températures, de variations entre les transistors). Nous pouvons tout de même émettre des réserves sur trois points, le premier est due au surcroit des connexions et des transistors dans les circuits (due a la logique sur deux rails et aux circuits de synchronisation) et donc à l'augmentation de la superficie. Le second est le manque de concepteurs qualifiés dans le développement et la maintenance de ces systèmes, et enfin son incompatibilité avec les outils CAO (temps de développement amplifié). Mais je pense que, dans ce cas, le jeu en vaut la chandelle. Il est d'ailleurs intéressant de noter que la nouvelle architecture, entièrement numérique, d'IBM (Merolla et al. (2014)) implémentant un million de neurones et 256 millions de synapses, et qui est maintenant la référence dans le neuromorphique, utilise une logique asynchrone.

Dans un second temps, l'architecture statique des réseaux neuromorphique actuelle ne permet pas de mettre en place la modification permanente des connexions neuronales. Par exemple, après apprentissage, une connexion synaptique fortement dépréciée pourrait être supprimée ou plutôt réutilisée soit par le même neurone pré-synaptique, mais en se connectant à un nouveau neurone post-synaptique, soit par deux nouveaux neurones. Cela permettrait de ne conserver que les connexions utiles ainsi que de tester continuellement les relations entre les neurones. Un circuit numérique pourrait, je pense, permettre l'implémentation de ce comportement avec un adressage dynamique des connexions neuronales.

Dans un troisième temps, j'opterais pour réaliser une étude sur l'ajout de connexions excitatrices latérales entre les neurones d'une même couche avec un apprentissage STDP. Les poids synaptiques de ces connexions seraient élevés comparé autres connexions du réseau. En effet, après une forte potentialisation, si une connexion est à son poids maximal alors elle pourra déclencher le neurone post-synaptique. On pourra alors, d'après moi, permettre de mettre en place une notion d'ordre de déclenchement, et donc implémenter l'apprentissage d'un mouvement, d'une trajectoire et faire de la prédiction. Une règle STDP similaire à celles présentées dans la section 1.5.1 figure 1.14 permettrait de mettre en place cet apprentissage. Par exemple, si le neurone A à tendance à se déclencher avant le neurone B, alors leur connexion s'intensifie, et après apprentissage, la simple activation du neurone A pourra déclencher à son tour le neurone B. Si l'on considère que les neurones A,B,C, etc. correspondent à l'emplacement d'une balle dans l'aire visuelle, alors ce mécanisme permettra de prédire l'emplacement futur de cette balle.

Pour terminer sur une note personnelle, je dirais que si l'on prend en considération le jeune degré de développement des dispositifs memristifs ainsi que la direction de leur développement par les industriels qui n'est non pas axé vers le neuromorphique, mais plutôt vers le remplacement de la mémoire RAM conventionnelle, alors je suis persuadé que les dispositifs memristifs seront la clé essentielle au développement de systèmes VLSI neuromorphiques dédiés, robustes, compacts, performants et hautement parallèles.

Annexes

Table 1 – Table de vérité de la logique contrôlant les portes de transmission d'une anode et d'une cathode dans le crossbar.

	MODE				Anode				Cathode			
SEL	<2>	<1>	<0>	state	Vread	Gnd	Vset	Vmid	Vreset	Gnd	Pad	Vmid
0	0	0	0	gnd	0	1	0	0	0	1	0	0
0	0	0	1	read	0	1	0	0	0	1	0	0
0	0	1	0	set	0	0	0	1	0	0	0	1
0	0	1	1	reset	0	0	0	1	0	0	0	1
0	1	0	0	Hset	0	0	0	1	0	0	0	1
0	1	0	1	Hreset	0	0	0	1	0	0	0	1
0	1	1	0	Lset	0	0	0	1	0	0	0	1
0	1	1	1	Lreset	0	0	0	1	0	0	0	1
1	0	0	0	gnd	0	1	0	0	0	1	0	0
1	0	0	1	read	1	0	0	0	0	0	1	0
1	0	1	0	set	0	0	1	0	0	1	0	0
1	0	1	1	reset	0	1	0	0	1	0	0	0
1	1	0	0	Hset	0	0	1	0	0	0	0	1
1	1	0	1	Hreset	0	0	0	1	1	0	0	0
1	1	1	0	Lset	0	0	0	1	0	1	0	0
1	1	1	1	Lreset	0	1	0	0	0	0	0	1

Table 2 – Table de vérité de la logique contrôlant les portes de transmission d'une anode, d'une cathode et d'une ligne de grilles dans la matrice.

	MODE			Anode				Cathode			grille
SEL	<1>	<0>	state	Vread	Gnd	Vreset	Vset	Vreset	Gnd	Pad	Vg
0	0	0	gnd	0	1	0	0	0	1	0	0
0	0	1	read	0	1	0	0	0	1	0	0
0	1	0	reset	0	0	1	0	0	1	0	0
0	1	1	set	0	1	0	0	0	1	0	0
1	0	0	gnd	0	1	0	0	0	1	0	0
1	0	1	read	1	0	0	0	0	0	1	1
1	1	0	reset	0	1	0	0	1	0	0	1
1	1	1	set	0	0	0	1	0	1	0	1

Gamrat et al. 2014 Roclin et al. 2013

Listing 1 – code VHDL du module customisé présent dans le FPGA lors du test de la puce.

```
1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3  use IEEE.NUMERIC_STD.ALL;
4
5  entity cbram is
6          port(
7                  clk: in STD_LOGIC;
8                  rst: in STD_LOGIC;
9                  Conv_done:in STD_LOGIC;
10                 var_tmp_low:in STD_LOGIC_VECTOR(31 downto 0);
11                 var_tmp_up : in STD_LOGIC_VECTOR(31 downto 0);
12                 cmd: in STD_LOGIC_VECTOR(14 downto 0);
13                 MODE: out STD_LOGIC_VECTOR(2 downto 0);
14                 WSEL: out STD_LOGIC_VECTOR(3 downto 0);
15                 BSEL: out STD_LOGIC_VECTOR(3 downto 0);
16                 PAD_MUX: out STD_LOGIC_VECTOR(1 downto 0);
17                 SELEC: out STD_LOGIC_VECTOR(1 downto 0);
18                 SEL: out STD_LOGIC;
19                 finished: out STD_LOGIC;
20                 conv_now: out STD_LOGIC;
21                 cmd_read: out STD_LOGIC_VECTOR(14 downto 0);
22                 state: out STD_LOGIC_VECTOR(3 downto 0)
23         );
24 end cbram;
25
26 architecture behavioral of cbram is
27
28 type state_type is (s0,s1,s2,s3,s4);
29 type t_regs is record
30                 state: state_type;
31                 counter_down: integer;
32                 counter_up: integer;
33 end record;
34
35 signal current_s,next_s: t_regs;
36 begin
37 prorun: process (current_s,cmd,var_tmp_up,Conv_done,var_tmp_low)
38         variable v:t_regs;
39         begin
40                 v:= current_s;
41                 case current_s.state is
42                         when s0 =>--wait a command
43                                 v.state:=s0;
44                                 if (cmd(14)='1' OR cmd(13)='1' OR cmd(12)='1') then
45                                         v.state:=s1;
46                                         v.counter_down:=to_integer(unsigned(var_tmp_low));
47                                 end if;
48
49                         when s1=>--send all signals except MODE during time "var_tmp_low"
50                                 v.counter_down:=v.counter_down-1;
51                                 if (v.counter_down = 0) then
52                                         v.state:=s2;
53                                         v.counter_up := to_integer(unsigned(var_tmp_up));
54                                 end if;
55
56                         when s2=> --send MODE during time "var_tmp_up"
57                            if (v.counter_up > 0) then
58                                         v.counter_up:=v.counter_up-1;
59                                 else
60                                         if ((cmd(14 downto 12)="001" AND Conv_done='1') OR (cmd(14 downto 12)/="
                                               001")) then
61                                           v.state:=s3;
62                                           v.counter_down:=to_integer(unsigned(var_tmp_low));
63                                         end if;
64                                 end if;
65
66                         when s3=>-- clear MODE and wait time "var_tmp_low"
67                                 v.counter_down:=v.counter_down-1;
68                                 if (v.counter_down = 0) then
69                                         v.state:=s4;
70                                 end if;
71
72                         when s4=>--wait for no command
73                                 if (cmd="000000000000000") then
74                                         v.state:=s0;
75                                 end if;
76                 end case;
77                 next_s<=v;
78 end process;
79
80 CombiOuts: process(current_s,cmd,var_tmp_up,Conv_done,var_tmp_low)
81                 begin
82                 MODE <= (others=>'0');
83                 WSEL <= (others=>'0');
84                 BSEL <= (others=>'0');
85                 PAD_MUX <= (others=>'0');
86                 SELEC <= (others=>'0');
87                 SEL <= '0';
88                 finished <='0';
89                 conv_now<='0';
90                 cmd_read<=cmd;
91                 state<="1111";
92
```

```
93                  case current_s.state is
94                          when s0 =>--wait a command
95
96                          when s1=>--send all signals except MODE during time "var_tmp_low"
97                                  WSEL <= cmd(11 downto 8);
98                                  BSEL <= cmd(7 downto 4);
99                                  PAD_MUX <= cmd(3 downto 2);
100                                 SELEC <= cmd(1 downto 0);
101                                 SEL <= '0';
102
103                         when s2=>--send MODE during time "var_tmp_up"
104                                 WSEL <= cmd(11 downto 8);
105                                 BSEL <= cmd(7 downto 4);
106                                 PAD_MUX <= cmd(3 downto 2);
107                                 SELEC <= cmd(1 downto 0);
108                                 SEL <= '0';
109                                 if (cmd(14 downto 12)="001") then --read
110                                         MODE <= "001";
111                                         conv_now<='1';
112                                 if (cmd(14 downto 12)="010") then
113                                         MODE <= "010";--set
114                                 elsif (cmd(14 downto 12)="011") then
115                                         MODE <= "011";--reset
116                                 end if;
117
118                         when s3=>-- clear MODE and wait time "var_tmp_low"
119                                 MODE <= "000";
120                                 WSEL <= cmd(11 downto 8);
121                                 BSEL <= cmd(7 downto 4);
122                                 PAD_MUX <= cmd(3 downto 2);
123                                 SELEC <= cmd(1 downto 0);
124                                 SEL <= '0';
125
126                         when s4=>--wait for no command
127                                 WSEL <= (others=>'0');
128                                 BSEL <= (others=>'0');
129                                 PAD_MUX <= (others=>'0');
130                                 SELEC <= (others=>'0');
131                                 SEL <= '0';
132                                 finished <='1';
133
134                 end case;
135 end process;
136
137 process (clk,rst)
138 begin
139         if (rst='0') then
140                 current_s.state <= s0; --default state on reset.
141                 current_s.counter_up<=0;
142                 current_s.counter_down<=0;
143         elsif (rising_edge(clk)) then
144                 current_s <= next_s; --state change.
145         end if;
146 end process;
147 end architecture;
```

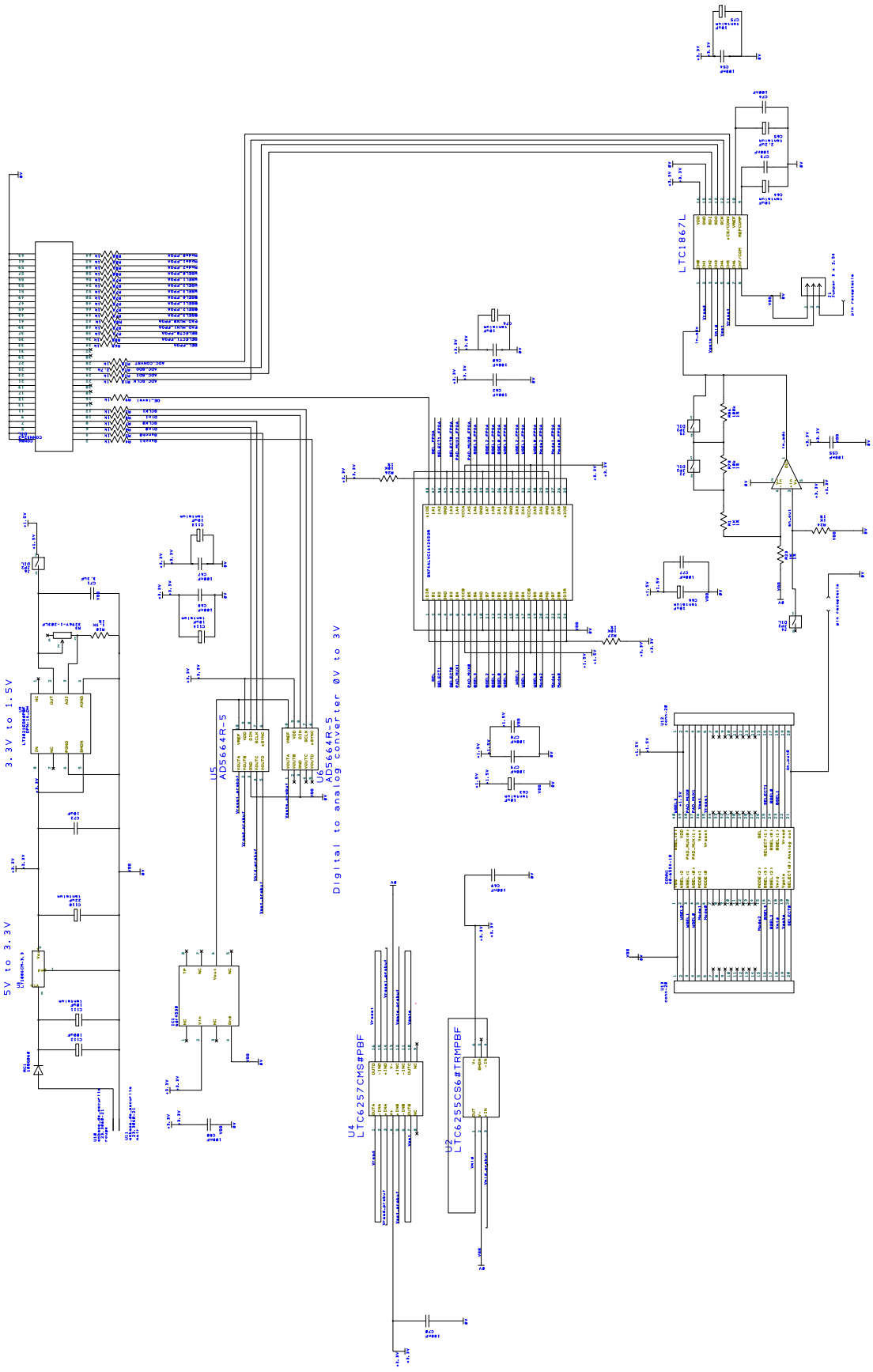


FIGURE 1 – PCB de test pour la puce neuromorphique. Les principaux composant sont : 5 DAC, 1 ADC, et 16 convertisseur de niveaux.

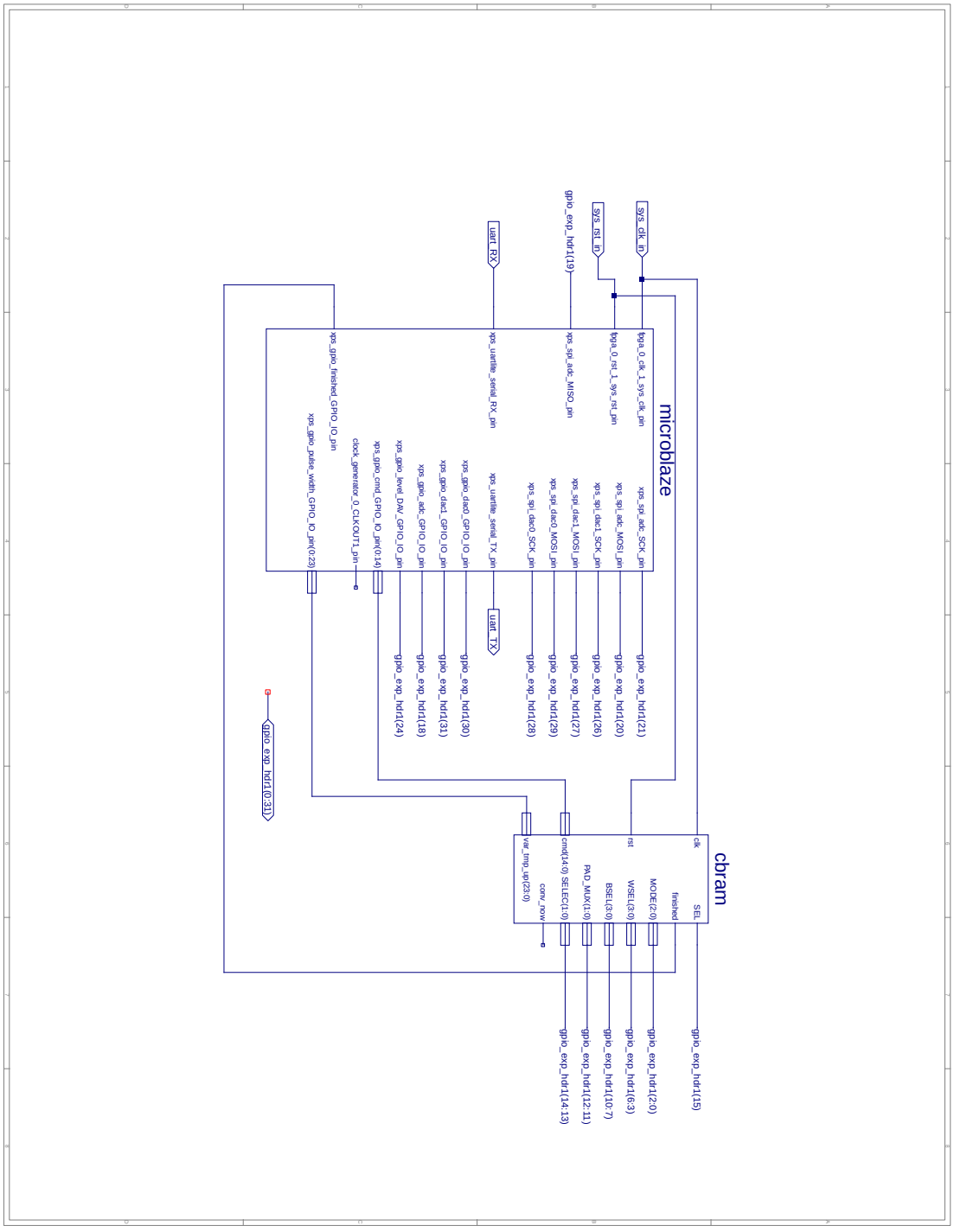


FIGURE 2 – PCB de test pour la puce neuromorphique. Les principaux composant sont : 5 DAC, 1 ADC, et 16 convertisseur de niveaux

Publications personnelles

Actes de conférences

O. Bichler, D. Roclin, C. Gamrat, and D. Querlioz. Design exploration methodology for memristor-based spiking neuromorphic architectures with the xnet event-driven simulator. In *Nanoscale Architectures (NANOARCH), 2013 IEEE/ACM International Symposium on*, pages 7–12, July 2013. doi:10.1109/NanoArch.2013.6623029.

D. Roclin, O. Bichler, C. Gamrat, S. Thorpe, and J.-O. Klein. Design study of efficient digital order-based stdp neuron implementations for extracting temporal features. In *Neural Networks (IJCNN), The 2013 International Joint Conference on*, pages 1–7, Aug 2013. doi:10.1109/IJCNN.2013.6707071.

D. Roclin, O. Bichler, C. Gamrat, and J.-O. Klein. Sneak paths effects in cbram memristive devices arrays for spiking neural networks. In *Nanoscale Architectures (NANOARCH), 2014 IEEE/ACM International Symposium on*, pages 13–18, July 2014. doi:10.1109/NANOARCH.2014.6880501.

Communications

C. Gamrat, D. Roclin, O. Bichler, M. Suri, D. Querlioz, and J.-O. Klein. Designing neuromorphic cicuits with memristive technologies. In *MemTDAC 1' (HIPEAC'14)*, 2014.

D. Roclin, O. Bichler, E. Vianello, M. Reyboz, M. Suri, D. Querlioz, C. Gamrat, and J.-O. Klein. Cbram as synapses for neuromorphic engineering. In *NANO-Saclay Nanoelectronics 2013*, 2013.

Bibliographie

Digest of Technical Papers, February 2013. IEEE. ISBN 978-1-4673-4515-6.

2014. URL http://www.institutdeloeil.com/traitement-de-loeil/anatomie-de-loeil.html.

K. Abe, M. Tendulkar, J. Jameson, P. B. Griffin, K. Nomura, S. Fujita, and Y. Nishi. Ultra-high bandwidth memory with 3d-stacked emerging memory cells. In *Integrated Circuit Design and Technology and Tutorial, 2008. ICICDT 2008. IEEE International Conference on*, pages 203–206, June 2008. doi:10.1109/ICICDT.2008.4567279.

J. Anderson and E. Rosenfeld. *Talking Nets : An Oral History of Neural Networks*. Bradford Books. MIT Press, 2000. ISBN 9780262511117. URL http://books.google.fr/books?id=-l-yim2lNRUC.

J. V. Arthur and K. Boahen. Learning in silicon : Timing is everything. In *NIPS'05*, pages –1–1, 2005.

M. Balakrishnan, S. Thermadam, M. Mitkova, and M. Kozicki. A low power non-volatile memory element based on copper in deposited silicon oxide. In *Non-Volatile Memory Technology Symposium, 2006. NVMTS 2006. 7th Annual*, pages 104–110, Nov 2006. doi:10.1109/NVMT.2006.378887.

M. Bear, B. Connors, and M. Paradiso. *Neuroscience : Exploring the Brain*. Lippincott Williams & Wilkins, 2007. ISBN 9780781760034.

R. Benenson. Classification datasets results. URL http://rodrigob.github.io/are_we_there_yet/build/classification_datasets_results.html.

G. Q. Bi and M. M. Poo. Synaptic modifications in cultured hippocampal neurons : dependence on spike timing, synaptic strength, and postsynaptic cell type. *J. Neurosci.*, 18(24) :10464–10472, Dec 1998.

O. Bichler. *Contribution à la conception d'architecture de calcul auto-adaptative intégrant des nanocomposants neuromorphiques et applications potentielles*. PhD thesis, Université Paris Sud-Paris XI, 2012.

O. Bichler, D. Querlioz, S. Thorpe, J. Bourgoin, and C. Gamrat. Unsupervised features extraction from asynchronous silicon retina through spike-timing-dependent plasticity. In *Neural Networks (IJCNN), The 2011 International Joint Conference on*, pages 859 –866, 31 2011-aug. 5 2011. doi:10.1109/IJCNN.2011.6033311.

O. Bichler, D. Querlioz, S. J. Thorpe, J.-P. Bourgoin, and C. Gamrat. Extraction of temporally correlated features from dynamic vision sensors with spike-timing-dependent plasticity. *Neural Networks*, 32(0) :339 – 348, 2012a. ISSN 0893-6080. doi:http ://dx.doi.org/10.1016/j.neunet.2012.02.022.

O. Bichler, M. Suri, D. Querlioz, D. Vuillaume, B. DeSalvo, and C. Gamrat. Visual pattern extraction using energy-efficient 2-pcm synapse neuromorphic architecture. *Electron Devices, IEEE Transactions on*, 59(8) :2206–2214, Aug 2012b. ISSN 0018-9383. doi:10.1109/TED.2012.2197951.

O. Bichler, C. Gamrat, and D. QUERLIOZ. Method for non-supervised learning in an artificial neurone network based on memristive nanodevices, and artificial neurone network implementing said method, 2013a. WO Patent App. PCT/EP2012/062,420.

O. Bichler, D. Roclin, C. Gamrat, and D. Querlioz. Design exploration methodology for memristor-based spiking neuromorphic architectures with the xnet event-driven simulator. In *Nanoscale Architectures (NANOARCH), 2013 IEEE/ACM International Symposium on*, pages 7–12, July 2013b. doi:10.1109/NanoArch.2013.6623029.

C. M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, Inc., New York, NY, USA, 1995. ISBN 0198538642.

K. Boahen. Point-to-point connectivity between neuromorphic chips using address events. *Circuits and Systems II : Analog and Digital Signal Processing, IEEE Transactions on*, 47(5) :416–434, May 2000. ISSN 1057-7130. doi:10.1109/82.842110.

K. A. Boahen. Communicating neuronal ensembles between neuromorphic chips. In T. Lande, editor, *Neuromorphic Systems Engineering*, volume 447 of *The Springer International Series in Engineering and Computer Science*, pages 229–259. Springer US, 1998. ISBN 978-0-7923-8158-7. doi:10.1007/978-0-585-28001-1_11.

R. Bruchhaus, M. Honal, R. Symanczyk, and M. Kund. Selection of optimized materials for cbram based on ht-xrd and electrical test results. *Journal of The Electrochemical Society*, 156(9) :H729–H733, 2009. doi:10.1149/1.3160570.

N. Caporale and Y. Dan. Spike timing-dependent plasticity : a Hebbian learning rule. *Annu. Rev. Neurosci.*, 31 :25–46, 2008.

G. Chevrier. Potentiel d' action, 2014. URL `http://bio.m2osw.com/gcartable/systeme%20nerveux/potentiels_d_action.htm`.

L. Chua. Memristor-the missing circuit element. *Circuit Theory, IEEE Transactions on*, 18(5) :507–519, Sep 1971. ISSN 0018-9324. doi:10.1109/TCT.1971.1083337.

D. C. Cireşan, U. Meier, L. M. Gambardella, and J. Schmidhuber. Deep, big, simple neural nets for handwritten digit recognition. *Neural Comput.*, 22(12) :3207–3220, Dec. 2010. ISSN 0899-7667. doi:10.1162/NECO_a_00052.

D. Ciresan, U. Meier, L. Gambardella, and J. Schmidhuber. Convolutional neural network committees for handwritten character classification. In *Document Analysis and Recognition (ICDAR), 2011 International Conference on*, pages 1135–1139, Sept 2011. doi:10.1109/ICDAR.2011.229.

C. A. Curcio, K. R. Sloan, R. E. Kalina, and A. E. Hendrickson. Human photoreceptor topography. *Journal of Comparative Neurology*, 292(4) :497–523, Feb. 1990. doi:10.1002/cne.902920402.

N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893 vol. 1, June 2005. doi:10.1109/CVPR.2005.177.

R. David, E. Williams, G. de Tremiolles, and P. Tannhof. Description and practical uses of ibm zisc036. volume 3728, pages 198–211, 1999. doi:10.1117/12.343038.

P. Dayan and L. Abbott. *Theoretical Neuroscience : Computational And Mathematical Modeling of Neural Systems*. Computational Neuroscience. Massachusetts Institute of Technology Press, 2005. ISBN 9780262541855.

C. Diorio, P. Hasler, B. Minch, and C. Mead. Floating-gate mos synapse transistors. In T. Lande, editor, *Neuromorphic Systems Engineering*, volume 447 of *The Springer International Series in Engineering and Computer Science*, pages 315–337. Springer US. ISBN 978-0-7923-8158-7. doi:10.1007/978-0-585-28001-1_14.

C. Diorio, P. Hasler, B. Minch, and C. Mead. A single-transistor silicon synapse. *Electron Devices, IEEE Transactions on*, 43(11) :1972–1980, Nov 1996. ISSN 0018-9383. doi:10.1109/16.543035.

S. Déthiollaz, 2014. URL http://web.expasy.org/prolune/dossiers/010/.

DVS128 Dynamic Vision Sensor Silicon Retina data, jan 2011. URL http://sourceforge.net/apps/trac/jaer/wiki/AER%20data.

J. G. Elias, D. P. M. Northmore, and W. Westerman. An analog memory circuit for spiking silicon neurons. *Neural Comput.*, 9(2) :419–440, Feb. 1997. ISSN 0899-7667. doi:10.1162/neco.1997.9.2.419.

B. N. Engel, J. Akerman, B. Butcher, R. Dave, M. DeHerrera, M. Durlam, G. Grynkewich, J. Janesky, S. Pietambaram, N. Rizzo, J. Slaughter, K. Smith, J. J. Sun, and S. Tehrani. A 4-mb toggle mram based on a novel bit and switching method. *Magnetics, IEEE Transactions on*, 41(1) :132–136, Jan 2005. ISSN 0018-9464. doi:10.1109/TMAG.2004.840847.

T.-N. Fang, S. Kaza, S. Haddad, A. Chen, Y.-C. Wu, Z. Lan, S. Avanzino, D. Liao, C. Gopalan, S. Choi, S. Mahdavi, M. Buynoski, Y. Lin, C. Marrian, C. Bill, M. VanBuskirk, and M. Taguchi. Erase mechanism for copper oxide resistive switching memory cells with nickel electrode. In *Electron Devices Meeting, 2006. IEDM '06. International*, Dec 2006. doi:10.1109/IEDM.2006.346731.

R. Fitzhugh. Impulses and Physiological States in Theoretical Models of Nerve Membrane. *Biophys. J.*, 1(6) :445–466, Jul 1961.

D. Frohman Bentchkowsky. Memory behavior in a floating gate avalanche injection mos (famos) structure. *Applied Physics Letters*, 18(8) :332–334, 1971. doi:http ://dx.doi.org/10.1063/1.1653685.

K. Fukushima. Neocognitron : A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, 36(4) : 193–202, 1980. ISSN 0340-1200. doi:10.1007/BF00344251.

S. Furber, D. Lester, L. Plana, J. Garside, E. Painkras, S. Temple, and A. Brown. Overview of the spinnaker system architecture. *Computers, IEEE Transactions on*, 62(12) :2454–2467, Dec 2013. ISSN 0018-9340. doi:10.1109/TC.2012.142.

W. Gallagher and S. Parkin. Development of the magnetic tunnel junction mram at ibm : From first junctions to a 16-mb mram demonstrator chip. *IBM Journal of Research and Development*, 50(1) :5–23, Jan 2006. ISSN 0018-8646. doi:10.1147/rd.501.0005.

D. Garbin, O. Bichler, E. Vianello, Q. Rafhay, C. Gamrat, L. Perniola, G. Ghibaudo, and B. DeSalvo. Variability-tolerant convolutional neural network for pattern recognition applications based on oxram synapses. In *Electron Devices Meeting (IEDM), 2014 IEEE International*, 2014.

A. H. Biological foundations - neuron communication, 2014. URL http://www.studyblue.com/notes/note/n/biological-foundations-neuron-communication-/deck/1025438.

D. Hammerstrom. A vlsi architecture for high-performance, low-cost, on-chip learning. In *Neural Networks, 1990., 1990 IJCNN International Joint Conference on*, pages 537–544 vol.2, June 1990. doi:10.1109/IJCNN.1990.137621.

J. Hasler and H. B. Marr. Finding a roadmap to achieve large neuromorphic hardware systems. *Frontiers in Neuroscience*, 7(118), 2013. ISSN 1662-453X. doi:10.3389/fnins.2013.00118.

D. O. Hebb. *The Organization of Behavior : A Neuropsychological Theory*. Wiley, New York, new ed edition, May 1949. ISBN 0805843000.

T. Hindo. Weight updating floating-gate synapse. *Electronics Letters*, 50(17) :1190–1191, Aug 2014. ISSN 0013-5194. doi:10.1049/el.2014.2039.

A. L. Hodgkin and A. F. Huxley. A quantitative description of membrane current and its application to conduction and excitation in nerve. *J. Physiol. (Lond.)*, 117(4) : 500–544, Aug 1952.

M. Holler, S. Tam, H. Castro, and R. Benson. An electrically trainable artificial neural network (etann) with 10240 'floating gate' synapses. In *Neural Networks, 1989. IJCNN., International Joint Conference on*, pages 191–196 vol.2, 1989. doi:10.1109/IJCNN.1989.118698.

Y. Hongxin, S. Luping, L. H. Koon, Z. Rong, and C. T. Chong. Endurance enhancement of elevated-confined phase change random access memory. *Japanese Journal of Applied Physics*, 51(2S) :02BD09, 2012. URL http://stacks.iop.org/1347-4065/51/i=2S/a=02BD09.

Y. Huai, Y. Zhou, I. Tudosa, R. Malmhall, R. Ranjan, and J. Zhang. Progress and outlook for stt-mram. In *Computer-Aided Design (ICCAD), 2011 IEEE/ACM International Conference on*, pages 235–235, Nov 2011. doi:10.1109/ICCAD.2011.6105332.

D. H. Hubel, T. N. Wiesel, D. Hubel, and T. Wiesel. Republication of The Journal of Physiology (1959) 148, 574-591 : Receptive fields of single neurones in the cat's striate cortex. 1959. *J. Physiol. (Lond.)*, 587(Pt 12) :2721–2732, Jun 2009.

S.-H. Hwang. Cmos image sensor : Current status and future perspectives, 2012.

C. T. Inc. Cm1k cognimem neuromorphic chip, 2014. URL http://www.cognimem.com/products/chips-and-modules/CM1K-Chip/.

G. Indiveri. Neuromorphic bistable VLSI synapses with spike-timing-dependent plasticity. In *Advances in Neural Information Processing Systems*, volume 15, pages 1091–1098, Cambridge, MA, USA, December 2002. MIT Press. URL http://ncs.ethz.ch/pubs/pdf/Indiveri02b.pdf.

G. Indiveri, E. Chicca, and R. Douglas. A vlsi array of low-power spiking neurons and bistable synapses with spike-timing dependent plasticity. *Neural Networks, IEEE Transactions on*, 17(1) :211–221, 2006.

M. Ito and M. Kano. Long-lasting depression of parallel fiber-purkinje cell transmission induced by conjunctive stimulation of parallel fibers and climbing fibers in the cerebellar cortex. *Neuroscience Letters*, 33(3) :253 – 258, 1982. ISSN 0304-3940. doi:http ://dx.doi.org/10.1016/0304-3940(82)90380-9.

ITRS. The international technology roadmap for semiconductors. 2013. URL `http://www.itrs.net/Links/2013ITRS/Summary2013.htm`.

J. Jameson, N. Gilbert, F. Koushan, J. Saenz, J. Wang, S. Hollmer, M. Kozicki, and N. Derhacobian. Quantized conductance in $ag/ges_2/w$ conductive-bridge memory cells. *Electron Device Letters, IEEE*, 33(2) :257–259, Feb 2012. ISSN 0741-3106. doi:10.1109/LED.2011.2177803.

J. Jameson, P. Blanchard, C. Cheng, J. Dinh, A. Gallo, V. Gopalakrishnan, C. Gopalan, B. Guichet, S. Hsu, D. Kamalanathan, D. Kim, F. Koushan, M. Kwan, K. Law, D. Lewis, Y. Ma, V. McCaffrey, S. Park, S. Puthenthermadam, E. Runnion, J. Sanchez, J. Shields, K. Tsai, A. Tysdal, D. Wang, R. Williams, M. Kozicki, J. Wang, V. Gopinath, S. Hollmer, and M. Van Buskirk. Conductive-bridge memory (cbram) with excellent high-temperature retention. In *Electron Devices Meeting (IEDM), 2013 IEEE International*, pages 30.1.1–30.1.4, Dec 2013. doi:10.1109/IEDM.2013.6724721.

D. S. Jeong, R. Thomas, R. S. Katiyar, J. F. Scott, H. Kohlstedt, A. Petraru, and C. S. Hwang. Emerging memories : resistive switching mechanisms and current status. *Reports on Progress in Physics*, 75(7) :076502, 2012.

S. H. Jo, T. Chang, I. Ebong, B. B. Bhadviya, P. Mazumder, and W. Lu. Nanoscale memristor device as synapse in neuromorphic systems. *Nano Letters*, 10(4) :1297–1301, 2010. doi:10.1021/nl904092h.

A. Joubert, B. Belhadj, O. Temam, and R. Heliot. Hardware spiking neurons design : Analog or digital ? In *Neural Networks (IJCNN), The 2012 International Joint Conference on*, pages 1–5, June 2012. doi:10.1109/IJCNN.2012.6252600.

Y.-B. Kim, S. R. Lee, D. Lee, C. B. Lee, M. Chang, J. H. Hur, M.-J. Lee, G.-S. Park, C.-J. Kim, U.-i. Chung, I.-K. Yoo, and K. Kim. Bi-layered rram with unlimited endurance and extremely uniform switching. In *VLSI Technology (VLSIT), 2011 Symposium on*, pages 52–53, June 2011.

M. Kozicki, M. Park, and M. Mitkova. Nanoscale memory elements based on solid-state electrolytes. *Nanotechnology, IEEE Transactions on*, 4(3) :331–338, May 2005. ISSN 1536-125X. doi:10.1109/TNANO.2005.846936.

A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. Burges, L. Bottou, and K. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

M. Kund, G. Beitel, C.-U. Pinnow, T. Rohr, J. Schumann, R. Symanczyk, K.-D. Ufert, and G. Muller. Conductive bridging ram (cbram) : an emerging non-volatile memory technology scalable to sub 20nm. In *Electron Devices Meeting, 2005. IEDM Technical Digest. IEEE International*, pages 754–757, Dec 2005. doi:10.1109/IEDM.2005.1609463.

E. A. Kurniawan, 2014. URL http://nbviewer.ipython.org/github/ekaakurniawan/3nb/blob/master/LifNeuron.ipynb?create=1.

S. Lai. Current status of the phase change memory and its future. In *Electron Devices Meeting, 2003. IEDM '03 Technical Digest. IEEE International*, pages 10.1.1–10.1.4, Dec 2003. doi:10.1109/IEDM.2003.1269271.

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11) :2278–2324, Nov 1998. ISSN 0018-9219. doi:10.1109/5.726791.

H. Lee, P. Chen, T. Y. Wu, Y. Chen, C. Wang, P. Tzeng, C. H. Lin, F. Chen, C. Lien, and M. J. Tsai. Low power and high speed bipolar switching with a thin reactive ti buffer layer in robust hfo2 based rram. In *Electron Devices Meeting, 2008. IEDM 2008. IEEE International*, pages 1–4, Dec 2008. doi:10.1109/IEDM.2008.4796677.

Y. Li, S. Lee, Y. Fong, F. Pan, T.-C. Kuo, J. Park, T. Samaddar, H. Nguyen, M. Mui, K. Htoo, T. Kamei, M. Higashitani, E. Yero, G. Kwon, P. Kliza, J. Wan, T. Kaneko, H. Maejima, H. Shiga, M. Hamada, N. Fujita, K. Kanebako, E. Tarn, A. Koh, I. Lu, C. Kuo, T. Pham, J. Huynh, Q. Nguyen, H. Chibvongodze, M. Watanabe, K. Oowada, G. Shah, B. Woo, R. Gao, J. Chan, J. Lan, P. Hong, L. Peng, D. Das, D. Ghosh, V. Kalluru, S. Kulkarni, R. Cernea, S. Huynh, D. Pantelakis, C.-M. Wang, and K. Quader. A 16gb 3b/ cell nand flash memory in 56nm with 8mb/s write rate. In *Solid-State Circuits Conference, 2008. ISSCC 2008. Digest of Technical Papers. IEEE International*, pages 506–632, Feb 2008. doi:10.1109/ISSCC.2008.4523279.

P. Lichtsteiner, C. Posch, and T. Delbruck. A 128x128 120 db 15 μs latency asynchronous temporal contrast vision sensor. *IEEE J JSSC*, 43(2) :566–576, 2008. ISSN 0018-9200. doi:10.1109/JSSC.2007.914337.

B. Linares-Barranco, E. Sanchez-Sinencio, A. Rodriguez-Vazquez, and J. Huertas. A cmos implementation of fitzhugh-nagumo neuron model. *Solid-State Circuits, IEEE Journal of*, 26(7) :956–965, Jul 1991. ISSN 0018-9200. doi:10.1109/4.92015.

S.-C. Liu. Analog vlsi circuits for short-term dynamic synapses. *EURASIP Journal on Advances in Signal Processing*, 2003(7) :596576, 2003. ISSN 1687-6180. doi:10.1155/S1110865703302094. URL http://asp.eurasipjournals.com/content/2003/7/596576.

T. Lømo. Frequency potentiation of excitatory synaptic activity in the dentate area of the hippocampal formation. *Acta Physiol. Scand*, 68(Suppl 277) :28+, 1966.

T. Lømo. The discovery of long-term potentiation. *Philosophical Transactions of the Royal Society of London. Series B : Biological Sciences*, 358(1432) :617–620, 2003. doi:10.1098/rstb.2002.1226.

D. Lowe. Object recognition from local scale-invariant features. In *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, volume 2, pages 1150–1157 vol.2, 1999. doi:10.1109/ICCV.1999.790410.

W. Maass. Networks of spiking neurons : The third generation of neural network models. *Neural Networks*, 10(9) :1659 – 1671, 1997. ISSN 0893-6080. doi:http ://dx.doi.org/10.1016/S0893-6080(97)00011-7.

M. Mahowald. *An Analog VLSI System for Stereoscopic Vision*. Kluwer Academic Publishers, Norwell, MA, USA, 1994. ISBN 0792394445.

M. Mahowald and L. Watts. A spike based learning neuron in analog vlsi. In *Advances in Neural Information Processing Systems 9*, pages 692–698. MIT Press, 1997.

M. A. Mahowald and C. Mead. The silicon retina. *j-SCI-AMER*, 264 (5) :76–82, May 1991. ISSN 0036-8733 (print), 1946-7087 (electronic). doi:http ://dx.doi.org/10.1038/scientificamerican0591-76.

H. Markram, W. Gerstner, and P. J. Sjöström. A history of spike-timing-dependent plasticity. *Frontiers in Synaptic Neuroscience*, 3(4), 2011. ISSN 1663-3563. doi:10.3389/fnsyn.2011.00004.

G. Marotta, A. Macerola, A. D'Alessandro, A. Torsi, C. Cerafogli, C. Lattaro, C. Musilli, D. Rivers, E. Sirizotti, F. Paolini, G. Imondi, G. Naso, G. Santin, L. Botticchio, L. De Santis, L. Pilolli, M. Gallese, M. Incarnati, M. Tiburzi, P. Conenna, S. Perugini, V. Moschiano, W. Di Francesco, M. Goldman, C. Haid, D. Di Cicco, D. Orlandi, F. Rori, M. Rossini, T. Vali, R. Ghodsi, and F. Roohparvar. A 3bit/cell 32gb nand flash memory at 34nm with 6mb/s program throughput and with dynamic 2b/cell blocks configuration mode for a program throughput increase up to 13mb/s. In *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2010 IEEE International*, pages 444–445, Feb 2010. doi:10.1109/ISSCC.2010.5433949.

T. Masquelier and S. J. Thorpe. Unsupervised learning of visual features through spike timing dependent plasticity. *PLoS Comput Biol*, 3(2) :e31, 02 2007. doi:10.1371/journal.pcbi.0030031.

N. Mauduit, M. Duranton, J. Gobert, and J.-A. Sirat. Lneuro 1.0 : a piece of hardware lego for building neural network systems. *Neural Networks, IEEE Transactions on*, 3(3) :414–422, May 1992. ISSN 1045-9227. doi:10.1109/72.129414.

W. S. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biology*, 5(4) :115–133, Dec. 1943. ISSN 0007-4985. doi:10.1007/bf02478259.

C. Mead. Adaptive retina. In C. Mead and M. Ismail, editors, *Analog VLSI Implementation of Neural Systems*, volume 80 of *The Kluwer International Series in Engineering and Computer Science*, pages 239–246. Springer US, 1989. ISBN 978-1-4612-8905-0. doi:10.1007/978-1-4613-1639-8_10.

C. A. Mead and M. Mahowald. A silicon model of early visual processing. *Neural Networks*, 1(1) :91 – 97, 1988. ISSN 0893-6080. doi:http ://dx.doi.org/10.1016/0893-6080(88)90024-X.

P. A. Merolla, J. V. Arthur, R. Alvarez-Icaza, A. S. Cassidy, J. Sawada, F. Akopyan, B. L. Jackson, N. Imam, C. Guo, Y. Nakamura, B. Brezzo, I. Vo, S. K. Esser, R. Appuswamy, B. Taba, A. Amir, M. D. Flickner, W. P. Risk, R. Manohar, and D. S. Modha. A million spiking-neuron integrated circuit with a scalable communication network and interface. *Science*, 345(6197) :668–673, 2014. doi:10.1126/science.1254642.

J. Nagumo, S. Arimoto, and S. Yoshizawa. An active pulse transmission line simulating nerve axon. *Proceedings of the IRE*, 50(10) :2061–2070, Oct 1962. ISSN 0096-8390. doi:10.1109/JRPROC.1962.288235.

I. P. Pavlov and G. V. Anrep. *Conditioned Reflexes : An Investigation of the Physiological Activity of the Cerebral Cortex*. Oxford University Press : Humphrey Milford, 1927.

T. Pfeil, T. C. Potjans, S. Schrader, W. Potjans, J. Schemmel, M. Diesmann, and K. Meier. Is a 4-bit synaptic weight resolution enough ? - constraints on enabling spike-timing dependent plasticity in neuromorphic hardware. *Frontiers in Neuroscience*, 6(90), 2012. ISSN 1662-453X. doi:10.3389/fnins.2012.00090.

A. Pirovano, A. Lacaita, A. Benvenuti, F. Pellizzer, and R. Bez. Electronic switching in phase-change memories. *Electron Devices, IEEE Transactions on*, 51(3) :452–459, March 2004. ISSN 0018-9383. doi:10.1109/TED.2003.823243.

C. Poultney, S. Chopra, and Y. Lecun. Efficient learning of sparse representations with an energy-based model. In *Advances in Neural Information Processing Systems (NIPS 2006*. MIT Press, 2006.

D. Purves. *Neuroscience*. Sinauer, Sunderland, Mass, 2008. ISBN 978-0878936977.

D. Purves and J. Coquery. *Neurosciences : Avec CD-Rom Sylvius*. Neurosciences & cognition. De Boeck Supérieur, 2005. ISBN 9782804147976.

D. Querlioz, O. Bichler, and C. Gamrat. Simulation of a memristor-based spiking neural network immune to device variations. In *Neural Networks (IJCNN), The 2011 International Joint Conference on*, pages 1775 –1781, 2011. doi:10.1109/IJCNN.2011.6033439.

D. Querlioz, O. Bichler, P. Dollfus, and C. Gamrat. Immunity to device variations in a spiking neural network with memristive nanodevices. *Nanotechnology, IEEE Transactions on*, 12(3) :288–295, May 2013. ISSN 1536-125X. doi:10.1109/TNANO.2013.2250995.

S. Z. Rahaman, S. Maikap, A. Das, A. Prakash, Y. H. Wu, C.-S. Lai, T.-C. Tien, W.-S. Chen, H.-Y. Lee, F. T. Chen, M.-J. Tsai, and L.-B. Chang. Enhanced nanoscale resistive switching memory characteristics and switching mechanism using high-ge-content ge0.5se0.5 solid electrolyte. *Nanoscale Res Lett*, 7(1) :614, 2012. ISSN 1556-276X.

S. Raoux, G. Burr, M. Breitwisch, C. Rettner, Y. Chen, R. Shelby, M. Salinga, D. Krebs, S.-H. Chen, H. L. Lung, and C. Lam. Phase-change random access memory : A scalable technology. *IBM Journal of Research and Development*, 52(4.5) :465–479, July 2008. ISSN 0018-8646. doi:10.1147/rd.524.0465.

M. Reyboz, S. Onkaraiah, G. Palma, E. Vianello, and L. Perniola. Compact model of a cbram cell in verilog-a. In *Non-Volatile Memory Technology Symposium (NVMTS), 2012 12th Annual*, pages 94–97, Oct 2013. doi:10.1109/NVMTS.2013.6632872.

F. Rieke, W. Bialek, D. Warland, and R. d. R. van Steveninck. *Spikes : Exploring the Neural Code*. Bradford book. MIT Press, 1999. ISBN 9780262681087.

M. Riesenhuber and T. Poggio. Hierarchical models of object recognition in cortex. *Nature Neuroscience*, 2 :1019–1025, 1999.

D. Roclin. Optimum transistor sizing of mtncl threshold gates for various design constraints. Master's thesis, University of Arkansas, Fayetteville, 2010.

F. Rosenblatt. Neurocomputing : Foundations of research. chapter The Perception : A Probabilistic Model for Information Storage and Organization in the Brain, pages 89–114. MIT Press, Cambridge, MA, USA, 1988. ISBN 0-262-01097-6.

T. Sakamoto, S. Kaeriyama, H. Sunamura, M. Mizuno, H. Kawaura, T. Hasegawa, K. Terabe, T. Nakayama, and M. Aono. A nonvolatile programmable solid electrolyte nanometer switch. In *Solid-State Circuits Conference, 2004. Digest of Technical Papers. ISSCC. 2004 IEEE International*, pages 290–529 Vol.1, Feb 2004. doi:10.1109/ISSCC.2004.1332708.

D. Schacter, D. T. Gilbert, and D. M. Wegner. *Psychology (2nd Edition)*. Worth, New York, 2011.

C. J. Schatz. The developing brain. *Scientific American*, 267(3) :60–67, 1992.

J. Schemmel, D. Brüderle, A. Grübl, M. Hock, K. Meier, and S. Millner. A wafer-scale neuromorphic hardware system for large-scale neural modeling. In *Circuits and Systems (ISCAS), Proceedings of 2010 IEEE International Symposium on*, pages 1947–1950, May 2010. doi:10.1109/ISCAS.2010.5536970.

C. Schindler, S. Thermadam, R. Waser, and M. Kozicki. Bipolar and unipolar resistive switching in cu-doped sio2. *Electron Devices, IEEE Transactions on*, 54(10) :2762–2768, Oct 2007. ISSN 0018-9383.

J. Schmidhuber. Multi-column deep neural networks for image classification. In *Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, CVPR '12, pages 3642–3649, Washington, DC, USA, 2012. IEEE Computer Society. ISBN 978-1-4673-1226-4.

S. Seo, M.-J. Lee, D. Seo, E. J. Jeoung, D. S. Suh, Y. S. Joung, I. Yoo, I. Hwang, S. H. Kim, I. Byun, J. S. Kim, J. Choi, and B. Park. Reproducible resistance switching in polycrystalline nio films. *Applied Physics Letters*, 85(23) :5655–5657, Dec 2004. ISSN 0003-6951. doi:10.1063/1.1831560.

R. Serrano-Gotarredona, M. Oster, P. Lichtsteiner, A. Linares-Barranco, R. Paz-Vicente, F. Gomez-Rodriguez, L. Camunas-Mesa, R. Berner, M. Rivas-Perez, T. Delbruck, S.-C. Liu, R. Douglas, P. Hafliger, G. Jimenez-Moreno, A. Ballcels, T. Serrano-Gotarredona, A. Acosta-Jimenez, and B. Linares-Barranco. Caviar : A 45k neuron, 5m synapse, 12g connects/s aer hardware sensory-processing-learning-actuating system for high-speed visual object recognition and tracking. *Neural Networks, IEEE Transactions on*, 20(9) :1417–1438, Sept 2009. ISSN 1045-9227. doi:10.1109/TNN.2009.2023653.

T. Serre and M. Riesenhuber. Realistic modeling of simple and complex cell tuning in the hmax model, and implications for invariant object recognition in cortex, 2004.

T. Serre, G. Kreiman, M. Kouh, C. Cadieu, U. Knoblich, and T. Poggio. A quantitative theory of immediate visual recognition. *PROG BRAIN RES*, pages 33–56, 2007.

P. Y. Simard, D. Steinkraus, and J. C. Platt. Best practices for convolutional neural networks applied to visual document analysis. Institute of Electrical and Electronics Engineers, Inc., August 2003.

K. Siu, V. Roychowdhury, and T. Kailath. *Discrete Neural Computation : A Theoretical Foundation*. Prentice-Hall information and system sciences series. Prentice Hall PTR, 1995. ISBN 9780133007084. URL http://books.google.fr/books?id=mKhQAAAAMAAJ.

S. C. Smith and J. Di. Designing asynchronous circuits using null convention logic (ncl). *Synthesis Lectures on Digital Circuits and Systems*, 4(1) :1–96, 2009. doi:10.2200/S00202ED1V01Y200907DCS023.

S. Song, K. D. Miller, L. F. Abbott, and N. G. Program. Competitive hebbian learning through spike-timing-dependent synaptic plasticity, 2000.

Sony. Annual report 2012, special feature ii, 2012.

V. Srinivasan, G. Serrano, C. Twigg, and P. Hasler. A floating-gate-based programmable cmos reference. *Circuits and Systems I : Regular Papers, IEEE Transactions on*, 55 (11) :3448–3456, Dec 2008. ISSN 1549-8328. doi:10.1109/TCSI.2008.925351.

D. B. Strukov, G. S. Snider, D. R. Stewart, and R. S. Williams. The missing memristor found. *Nature*, 453(7191) :80–83, May 2008. ISSN 0028-0836. doi:10.1038/nature06932.

M. Suri, O. Bichler, D. Querlioz, O. Cueto, L. Perniola, V. Sousa, D. Vuillaume, C. Gamrat, and B. DeSalvo. Phase change memory as synapse for ultra-dense neuromorphic systems : Application to complex visual pattern extraction. In *Electron Devices Meeting (IEDM), 2011 IEEE International*, pages 4.4.1–4.4.4, Dec 2011. doi:10.1109/IEDM.2011.6131488.

M. Suri, D. Garbin, O. Bichler, D. Querlioz, D. Vuillaume, C. Gamrat, and B. DeSalvo. Impact of pcm resistance-drift in neuromorphic systems and drift-mitigation strategy. In *Nanoscale Architectures (NANOARCH), 2013 IEEE/ACM International Symposium on*, pages 140–145, July 2013a. doi:10.1109/NanoArch.2013.6623059.

M. Suri, D. Querlioz, O. Bichler, G. Palma, E. Vianello, D. Vuillaume, C. Gamrat, and B. DeSalvo. Bio-inspired stochastic computing using binary cbram synapses. *Electron Devices, IEEE Transactions on*, 60(7) :2402–2409, July 2013b. ISSN 0018-9383. doi:10.1109/TED.2013.2263000.

S. Tehrani, J. Slaughter, E. Chen, M. Durlam, J. Shi, and M. DeHerren. Progress and outlook for mram technology. *Magnetics, IEEE Transactions on*, 35(5) :2814–2819, Sep 1999. ISSN 0018-9464. doi:10.1109/20.800991.

S. Thorpe. Spike arrival times : A highly efficient coding scheme for neural networks. *Parallel processing in neural systems*, pages 91–94, 1990.

S. Thorpe, D. Fize, and C. Marlot. Speed of processing in the human visual system. *Nature*, 381 :520, 1996.

S. J. Thorpe and M. Fabre-Thorpe. Seeking categories in the brain. *Science*, 291(5502) : 260–263, 2001. doi:10.1126/science.1058249.

S. J. Thorpe, R. Guyonneau, N. Guilbaud, J.-M. Allegraud, and R. VanRullen. SpikeNet : real-time visual processing with one spike per neuron. *Neurocomputing*, 58–60 (0) :857 – 864, 2004. ISSN 0925-2312. doi:10.1016/j.neucom.2004.01.138.

E. B. Tomaso Poggio. Generalization in vision and motor control, 2004.

K. Tsunoda, Y. Fukuzumi, J. R. Jameson, Z. Wang, P. B. Griffin, and Y. Nishi. Bipolar resistive switching in polycrystalline tio2 films. *Applied Physics Letters*, 90(11) : 113501, 2007. doi:http ://dx.doi.org/10.1063/1.2712777.

D. Van Essen. Organization of visual areas in macaque and human cerebral cortex. *In The Visual Neurosciences, L.M. Chalupaand J.S. Werner, eds. (Boston : Bradford Books)*, 26(7) :507–521, Jul 2003.

l. Vázquez and E. Antelo. Implementation of the exponential function in a floating-point unit. *Journal of VLSI signal processing systems for signal, image and video technology*, 33 :125–145, 2003. ISSN 0922-5773. doi:10.1023/A :1021102104078.

L. Wan, M. Zeiler, S. Zhang, Y. L. Cun, and R. Fergus. Regularization of neural networks using dropconnect. In *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, pages 1058–1066, 2013.

B. A. Wandell, S. O. Dumoulin, and A. A. Brewer. Visual field maps in human cortex. *Neuron*, 56(2) :366 – 383, 2007. ISSN 0896-6273. doi:http ://dx.doi.org/10.1016/j.neuron.2007.10.012.

R. Waser and M. Aono. Nanoionics-based resistive switching memories. *Nature materials*, 6(11) :833–840, 2007.

H.-S. Wong, S. Raoux, S. Kim, J. Liang, J. P. Reifenberg, B. Rajendran, M. Asheghi, and K. E. Goodson. Phase change memory. *Proceedings of the IEEE*, 98(12) :2201–2227, Dec 2010a. ISSN 0018-9219. doi:10.1109/JPROC.2010.2070050.

H.-S. Wong, H.-Y. Lee, S. Yu, Y.-S. Chen, Y. Wu, P.-S. Chen, B. Lee, F. Chen, and M.-J. Tsai. Metal-oxide rram. *Proceedings of the IEEE*, 100(6) :1951–1970, June 2012. ISSN 0018-9219. doi:10.1109/JPROC.2012.2190369.

H. S. P. Wong, S. Raoux, S. Kim, J. Liang, J. P. Reifenberg, B. Rajendran, M. Asheghi, and K. E. Goodson. Phase change memory. *Proceedings of the IEEE*, 98(12) :2201–2227, Dec 2010b. ISSN 0018-9219. doi:10.1109/JPROC.2010.2070050.

Y. Yang, P. Gao, S. Gaba, T. Chang, X. Pan, and W. Lu. Observation of conducting filament growth in nanoscale resistive memories. *Nature communications*, 3 :732, 2012.

M. Yasunaga, N. Masuda, M. Yagyu, M. Asai, M. Yamada, and A. Masaki. Design, fabrication and evaluation of a 5-inch wafer scale neural network lsi composed on 576 digital neurons. In *Neural Networks, 1990., 1990 IJCNN International Joint Conference on*, pages 527–535 vol.2, June 1990. doi:10.1109/IJCNN.1990.137618.

S. Yu and H. S. P. Wong. Modeling the switching dynamics of programmable-metallization-cell (pmc) memory and its application as synapse device for a neuromorphic computation system. In *Electron Devices Meeting (IEDM), 2010 IEEE International*, pages 22.1.1–22.1.4, Dec 2010. doi:10.1109/IEDM.2010.5703410.

S. Yu and H. S. P. Wong. Compact modeling of conducting-bridge random-access memory (cbram). *Electron Devices, IEEE Transactions on*, 58(5) :1352–1360, May 2011. ISSN 0018-9383. doi:10.1109/TED.2011.2116120.